Linguistisch unterstütztes Redigieren: Konzept und exemplarische Umsetzung basierend auf interaktiven computerlinguistischen Ressourcen

Abstract

Schreiben und Redigieren stellen hohe kognitive Anforderungen an Autoren. Selbst publizierte Texte sind nie ganz fehlerfrei. Für viele Fehler kann man die Entstehung rekonstruieren: Funktionen in Textbearbeitungsprogrammen sind zeichenbasiert und berücksichtigen nicht die Elemente und Strukturen der jeweiligen verwendeten Sprache. Autoren müssen ihre Redigierabsichten in eine lange, komplexe Folge solcher zeichenbasierten Funktionen übersetzen.

Editoren für Programmierer hingegen bieten seit langem sprachspezifische Editierfunktionen, die auf den Elementen und Strukturen der verwendeten Programmiersprache operieren. Diese Funktionen tragen dazu bei, das Ändern von Programmcode zu erleichtern und Fehler zu vermeiden.

In dieser Arbeit übertragen wir das Prinzip solcher sprachspezifischen Funktionen in Programmiereditoren auf Funktionen für die Bearbeitung natürlichsprachlicher Texte. Wir entwickeln das Konzept der linguistisch unterstützten Redigierfunktionen unter Berücksichtigung aktueller Erkenntnisse der Schreibforschung. Wir definieren Informations-, Bewegungs- und Modifikationsfunktionen, die auf Elementen und Strukturen natürlicher Sprache operieren. Solche Funktionen sollen Autoren entlasten und helfen, typische Fehler zu vermeiden.

Sprachspezifische Funktionen beruhen auf Methoden zur Erkennung und Bestimmung relevanter Elemente und Strukturen. Wir verwenden dazu computerlinguistische Ressourcen zur morphologischen Analyse und Generierung und zur automatischen Wortartenbestimmung. Die Evaluation verfügbarer Ressourcen ergibt, dass die Situation für die Behandlung des Deutschen nicht so vielversprechend ist, wie ursprünglich angenommen und üblicherweise in der Literatur dargestellt.

Unsere prototypische Implementierung linguistisch unterstützter Redigierfunktionen für die Bearbeitung deutscher Texte zeigt die Möglichkeiten und Grenzen des Konzepts unter Berücksichtigung der Leistungsfähigkeit heute verfügbarer computerlinguistischer Ressourcen und der Eigenschaften des Deutschen.

(english abstract)

Composing, revising, and editing are highly demanding tasks. Even in polished and published texts from professional writers we can observe errors and mistakes. For many errors, we can infer how they came to be: Word processors offer character-based functions only. These functions do not take into account elements and structures of the language the author is using. Authors are thus forced to translate their high-level goals into long and complex sequences of low-level character-based functions. Both the translation process and

the execution of such sequences of functions are error-prone.

However, in text editors for programmers we find so-called language-aware editing functions. These functions operate on the elements and structures of a programming or mark-up language and help to avoid errors, as language-aware functions make revising and editing less tedious and error-prone.

We argue that the concept of language awareness can be transferred to writing natural language texts using word processors. We propose functions that take the structures of natural languages into consideration. We distinguish information functions, movement functions, and operations to support revising and editing. The design is based on current findings from writing research.

Language-aware editing functions rely on the recognition and categorization of relevant elements and structures with respect to a certain language. We use methods and resources from computational linguistics for morphological analysis and generation, and for part-of-speech tagging. When evaluating respective resources we face a rather disappointing situation: NLP resources for German are less suitable than assumed and less applicable for real-world applications than usually claimed in the literature.

Our prototypical implementation of language-aware functions for revising and editing of German texts serves as a proof of concept. The implementation illustrates opportunities and limits of current NLP resources for German.

Linguistisch unterstütztes Redigieren: Konzept und exemplarische Umsetzung basierend auf interaktiven computerlinguistischen Ressourcen

Abhandlung zur Erlangung der Doktorwürde
der Philosophischen Fakultät der Universität Zürich

Vorgelegt von
Cerstin Elisabeth Mahlow
aus Deutschland

Angenommen im
Frühjahrssemester 2011

auf Antrag von
Prof. Dr. Michael Hess
Dr. Michael Zock

Linguistisch unterstütztes Redigieren: Konzept und exemplarische Umsetzung basierend auf interaktiven computerlinguistischen Ressourcen

Design von Michael Piotrowski (mpreport).
Gesetzt mit LaTeX in Linotype Goudy Oldstyle und Helvetica.

Zusammenfassung

Schreiben und Redigieren stellen hohe kognitive Anforderungen an Autoren. Selbst publizierte Texte sind nie ganz fehlerfrei. Für viele Fehler kann man die Entstehung rekonstruieren: Funktionen in Textbearbeitungsprogrammen sind zeichenbasiert und berücksichtigen nicht die Elemente und Strukturen der jeweiligen verwendeten Sprache. Autoren müssen ihre Redigierabsichten in eine lange, komplexe Folge solcher zeichenbasierten Funktionen übersetzen.

Editoren für Programmierer hingegen bieten seit langem sprachspezifische Editierfunktionen, die auf den Elementen und Strukturen der verwendeten Programmiersprache operieren. Diese Funktionen tragen dazu bei, das Ändern von Programmcode zu erleichtern und Fehler zu vermeiden.

In dieser Arbeit übertragen wir das Prinzip solcher sprachspezifischen Funktionen in Programmiereditoren auf Funktionen für die Bearbeitung natürlichsprachlicher Texte. Wir entwickeln das Konzept der linguistisch unterstützten Redigierfunktionen unter Berücksichtigung aktueller Erkenntnisse der Schreibforschung. Wir definieren Informations-, Bewegungs- und Modifikationsfunktionen, die auf Elementen und Strukturen natürlicher Sprache operieren. Solche Funktionen sollen Autoren entlasten und helfen, typische Fehler zu vermeiden.

Sprachspezifische Funktionen beruhen auf Methoden zur Erkennung und Bestimmung relevanter Elemente und Strukturen. Wir verwenden dazu computerlinguistische Ressourcen zur morphologischen Analyse und Generierung und zur automatischen Wortartenbestimmung. Die Evaluation verfügbarer Ressourcen ergibt, dass die Situation für die Behandlung des Deutschen nicht so vielversprechend ist, wie ursprünglich angenommen und üblicherweise in der Literatur dargestellt.

Unsere prototypische Implementierung linguistisch unterstützter Redigierfunktionen für die Bearbeitung deutscher Texte zeigt die Möglichkeiten und Grenzen des Konzepts unter Berücksichtigung der Leistungsfähigkeit heute verfügbarer computerlinguistischer Ressourcen und der Eigenschaften des Deutschen.

Abstract

Composing, revising, and editing are highly demanding tasks. Even in polished and published texts from professional writers we can observe errors and mistakes. For many errors, we can infer how they came to be: Word processors offer character-based functions only. These functions do not take into account elements and structures of the language the author is using. Authors are thus forced to translate their high-level goals into long and complex sequences of low-level character-based functions. Both the translation process and the execution of such sequences of functions are error-prone.

However, in text editors for programmers we find so-called language-aware editing functions. These functions operate on the elements and structures of a programming or mark-up language and help to avoid errors, as language-aware functions make revising and editing less tedious and error-prone.

We argue that the concept of language awareness can be transferred to writing natural language texts using word processors. We propose functions that take the structures of natural languages into consideration. We distinguish information functions, movement functions, and operations to support revising and editing. The design is based on current findings from writing research.

Language-aware editing functions rely on the recognition and categorization of relevant elements and structures with respect to a certain language. We use methods and resources from computational linguistics for morphological analysis and generation, and for part-of-speech tagging. When evaluating respective resources we face a rather disappointing situation: NLP resources for German are less suitable than assumed and less applicable for real-world applications than usually claimed in the literature.

Our prototypical implementation of language-aware functions for revising and editing of German texts serves as a proof of concept. The implementation illustrates opportunities and limits of current NLP resources for German.

Danksagungen

Die vorliegende Dissertation entstand am Institut für Computerlinguistik der Universität Zürich. Für die Unterstützung möchte ich mich vor allem bei meinem Doktorvater und Institutsdirektor Michael Hess bedanken. Grosser Dank gebührt ebenso dem zweiten Referenten Michael Zock. Beide haben mich – vor allem in den letzten Monaten des Schreibens – motiviert und standen immer für Austausch und fachlichen Diskurs zur Verfügung.

Danken möchte ich Robert Dale, Yves Forkl, Richard Hamlet, Donald Norman, Daniel Perrin und Kerstin Severinson-Eklundh für fachliche Diskussionen und anregenden Austausch. Sven Grund und Christian Schorno danke ich für Unterstützung beim Projektmanagement. Kai-Uwe Carstensen, Yves Forkl, Stefan Lewandowski, Ursula Mahlow, Wolfgang Mahlow, Christian Schorno und Eva Strübin danke ich für das Korrekturlesen verschiedener Fassungen dieser Arbeit. Verbliebene Fehler gehen allein zu meinen Lasten.

Meinen Eltern danke ich für ihre Unterstützung und Ermunterung während all der Jahre. Nicht zuletzt möchte ich Michael Piotrowski danken für seine Geduld und sein Verständnis, für die oft nächtelangen Diskussionen zum Konzept der Arbeit, für das Teilen der Freude bei Aha-Erlebnissen und die Relativierung der Enttäuschung in Sackgassen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

1 Einleitung und Motivation

For many people, writing is painful and editing one's own prose is difficult, tedious and errorprone. It is often hard to see which parts of a document are difficult to read or how to transform a wordy sentence into a more concise one. It is even harder to discover that one overuses a particular linguistic construct.

—Linda Cherry, *Computer Aids for Writers*, 1981

Ein grosser Vorteil der Textproduktion am Computer liegt darin, dass fehlerhafte Wörter oder holprige Formulierungen, Auslassungsfehler oder gar ganze Textpassagen, die an der falschen Stelle stehen, im Handumdrehen verbessert und überarbeitet werden können.

—Jasmin Merz-Grötsch, *Schreiben als System. Band 1: Schreibforschung und Schreibdidaktik. Ein Überblick*, 2000

Viele wissenschaftliche Texte im Bereich Schreibforschung, Editoren[1] und Schreibunterstützung beginnen mit der Aussage, dass Schreiben eine *schwierige*, *anstrengende* und *fehleranfällige* Angelegenheit sei [etwa Alamargot und Chanquoy 2001, Cherry 1981, Daiute 1983, Fayol 1999, Negro und Chanquoy 1999, Puerta Melguizo et al. 2008a, Quinlan et al. 2009, Ransdell und Levy 1999], sodass Wengelin et al. ihren Aufsatz so eröffnen: «The most worn-out cliché within writing research is probably that writing *is a complex process*. Nevertheless it is true.» [Wengelin et al. 2010, S. 735]

1. Wir verwenden den Begriff *Editor* für *Computerprogramme*, die dem Erstellen und Bearbeiten von Text dienen – egal ob es sich um natürlichsprachlichen Text oder Programmcode handelt. Für *Personen*, die einen Text editieren, verwenden wir jeweils Bezeichnungen, die deutlich machen, dass es sich nicht um ein Programm handelt.

Schreiben ist ein Prozess: Aus einer Idee entwickeln sich erste Wortgruppen, aus Wortgruppen einzelne Sätze, aus Sätzen ganze Texte. Während des Schreibens unterliegt der entstehende Text ständigen Veränderungen. Schon die Erstfassung eines Textes kann permanentes Ändern an bereits Geschriebenem beinhalten.

Ist eine erste Fassung erstellt, werden häufig Meinungen von Kollegen und Lektoren eingeholt und deren Änderungsvorschläge in einer zweiten, dritten, vierten Fassung in den eigenen Text eingearbeitet. Aber auch die Arbeit ohne Feedback von aussen führt in der Regel zu einem endgültigen Text, der sich von seiner ersten Fassung stark unterscheidet. Die Überarbeitung betrifft strukturelle Aspekte auf der Ebene des Gesamttextes bis hinunter zu einzelnen Absätzen sowie Änderungen auf Satz- und Wortebene. Textteile werden umgestellt, einzelne Wörter oder Wortgruppen entfernt, durch andere ersetzt, hinzugefügt. Schreiben ist eine kreative Tätigkeit, die auch spielerische Aspekte enthält: Verschiedene Formulierungen werden probiert, Sätze und Wortgruppen umgestellt, neue Wörter «erfunden» etc., siehe auch [Sharples 1996a, 1999].

Schreiben setzt die Verwendung eines Werkzeugs zur Produktion bzw. Eingabe des Textes und das Vorhandensein eines Mediums, auf dem geschrieben wird, voraus. Beim Schreiben mit Papier und Stift oder der Schreibmaschine ist das Umstellen von Absätzen oder grösseren Textteilen schwierig. In der Regel wird der Autor eine erste Fassung des Textes produzieren, anschliessend notwendige Änderungen markieren und dann eine neue Fassung erstellen. Der Prozess der Verschriftlichung ist vom Prozess des Redigierens relativ klar zu trennen.

Moderne Textbearbeitungsprogramme bieten die Möglichkeit, mit den Funktionen Kopieren, Ausschneiden und Einfügen Textteile mit wenig Aufwand umzustellen. Die Arbeit mit diesen Werkzeugen führt dazu, auch grosse Änderungen schon *während* des Schreibens vorzunehmen. Die Prozesse der Texteingabe und des Redigierens vermischen sich.

Das verwendete Werkzeug (Stift, Schreibmaschine oder Textbearbeitungsprogramm) wie auch das Medium (Papier oder Bildschirm) beeinflussen also den Schreibprozess und auch das Schreibprodukt, den Text. Ermöglicht ein neues Werkzeug neue Arbeitsschritte – etwa das einfache Kopieren, Ausschneiden und Einfügen von Elementen –, können sich daraus auch neue Fehlerquellen ergeben: «[…] different kinds of errors that might be characteristic of writing with computers, having the same phrase, for example, at both the start and the end of a sentence because the writer forgot to delete one.» [Hawisher 1986, S. 16] Strategien zum Schreiben und Redigieren mit Papier und Stift sind daher nicht einfach auf das Arbeiten mit elektronischen Werkzeugen übertragbar.

Schreiben und Redigieren stellen hohe kognitive Anforderungen an den Autor. Redigier- und Editieroperationen sind komplex: Die freie Wortstellung im Deutschen und die Kongruenz innerhalb von Wortgruppen oder zwischen Satzgliedern erfordern es, stets den ganzen Satz und sogar daran anschliessende Sätze im Blick zu haben. Änderungen, die aus inhaltlichen oder rhetorischen Gründen an bestimmten Stellen vorgenommen werden, können syntaktische oder morphologische Änderungen an weiter entfernten Textstellen nach sich ziehen, die nicht im Fokus des Autors sind. Sie werden vom Autor oft übersehen

und selbst beim Korrekturlesen nicht wahrgenommen. So lassen sich typische Fehler in «fertigen»[2] Texten finden:

Sätze ohne Verb *«Dabei es zum vielfältige Arbeitsbereiche.»* (Beispiel A.14), *«Sie sich viel Zeit für den Bettenkauf.»* (Beispiel A.16), *«So wurden ab 2000 viele Einzelversuche und Pilotprojekte mit Portfolio-Arbeit in Schule, in Hochschule (v.a. in der Lehrerbildung) und in der beruflichen Bildung.»* (Beispiel A.20)

Sätze mit doppeltem finiten Verb *«Dabei ist Fälschung in der Forschung so alt ist wie die Wissenschaft selbst.»* (Beispiel A.55), *«In einem Steinbruch wurde nun eine männliche Leiche gefunden, nach Aussagen von Stolbunow ist es der Vermisste ist.»* (Beispiel A.58), *«Wir fliegen wie geplant am 30. Oktober nach Kairo fliegen.»* (Beispiel A.56), *«Momentan prüft die zuständige Staatsanwaltschaft München II prüft, ob in Kloster Ettal noch verfolgbare Straftaten vorliegen.»* (Beispiel A.73)

Wortwiederholungen *«Zum anderen bereite ich ein Projekt vor, in dem es um den Einsatz von Web 2.0 in dem es um die Weiterbildung älterer Arbeitnehmer geht.»* (Beispiel A.52), *«Die Büste repräsentiere Ägypten in Deutschland Ägypten und sei damit für die Kultur und Geschichte ihrer Heimat der beste Werbeträger.»* (Beispiel A.54), *«Pfeifend setzen sich die Rotoren der vierzig 40 Jahre alten elektrischen Schneeschleuder in Bewegung.»* (Beispiel A.67)

Falsche Wortstellung *«‹In der Bundesliga hat jetzt eine stabile zu Phase kommen›, fordert Rummenigge»* (Beispiel A.159), *«Von der grauhaarigen Frau sieht Christin nur zuckenden den Rücken.»* (Beispiel A.166), *«Wie ein Häufchen Elend saß Messi in der Kabine, weinte hemmungslos, war von niemandem zu trösten. Wortlos und leichenblass schlich er eineinhalb Stunden dem nach Spielende aus dem Stadion ‹Green Point› in Kapstadt.»* (Beispiel A.167)

Fehlende Kongruenz *«Das Projektteam innerhalb des vorliegenden Projekts setzt sich zusammen aus insgesamt fünf wissenschaftlichen Mitarbeitern (drei deutsche und zwei polnische Wissenschaftler), zwei Verwaltungsangestellten (anteilig; jeweils eine/ein Deutsche/Deutscher und eine/ein Pole/Polin), mehreren polnische und deutsche studentische Hilfskräften sowie Java- bzw. C/C++-Programmierer aus beiden Ländern.»* (Beispiel A.176), *«Bitte schreiben Sie Ihren Beitrag so, dass er für ein breites Publikum verständlich sind.»* (Beispiel A.177), *«…und hoffe, dass Sie einen schöne und erholsame Ferien hatten.»* (Beispiel A.181), *«Alle Studierenden dokumentieren und reflektieren die vorgegebene Kompetenzen.»* (Beispiel A.186)

Mehrfachfehler *«Und davon, dass zwei Lesben und ein Schwuler zu dritt ein Kind zu bekommen – während die Politik noch streitet, ob sie zu zweit eins adoptieren dürften.»* (Beispiel A.201), *«Mit einem Gewinn von 2,2 Milliarden Franken im ersten Quartal hat die UBS-Chef Oswald Grübel sogar seinen ehamligen Arbeitgeber schlagen.»* (Beispiel A.204)

Alle Beispiele stammen aus realen Texten aus Zeitungen, Büchern, E-Mails oder wissenschaftlichen Artikeln. Eine umfangreiche Fehlersammlung ist in Anhang A zu finden, wir kategorisieren dort die Fehler und geben Erklärungen zum möglichen Entstehen. Viele derartige Beispiele können darauf zurückgeführt werden, dass komplexe Editieraktionen durchgeführt wurden, die den Autor überforderten, wie Perrin erläutert:

2. Das heisst, in publizierten oder an andere Personen verschickten Texten.

> Bei komplexen Umbauten verlieren Schreibende leicht den Blick für das, was am Ende tatsächlich dasteht – sie sehen beim Nachlesen den Text in ihrem Kopf statt den Text auf dem Papier oder Bildschirm. [Perrin 2006b, S. 290]

Redigieroperationen können in sich komplex sein, etwa das konsistente Ersetzen eines Substantivs durch ein anderes oder die Änderung aller Verben, sodass sie neu im Präsens statt im Perfekt verwendet werden. Solche Redigieroperationen stellen hohe kognitive Anforderungen an den Autor und es schleichen sich leicht Fehler ein. Zudem können Editieraktionen *Seiteneffekte* haben, die dem Autor nicht bewusst sind, da seine kognitive Kapazität mit der eigentlichen Redigieroperation bereits ausgelastet ist. Die Auswirkungen einer Manipulation an Textstellen, die von der Stelle der eigentlichen Aktion entfernt sind, können in der Regel erst in einem weiteren Redigierdurchlauf gefunden werden.

Betrachten wir solche typischen Fehler in Texten genauer und berücksichtigen das Schreibwerkzeug – nämlich Textbearbeitungsprogramme –, können wir einen grossen Teil der Fehler auf das Design der verwendeten Schreibwerkzeuge zurückführen: Die vorhandenen Funktionen[3] in Textbearbeitungsprogrammen operieren auf einzelnen oder mehreren *Zeichen*, nicht auf *linguistischen Einheiten*. Diese Funktionen verhalten sich identisch, egal in welcher Sprache ein Text verfasst wird, da lediglich unspezifische Zeichenketten bearbeitet werden können. Im Unterschied dazu bieten Werkzeuge für Programmierer spezifische Funktionen für die jeweils verwendete Programmiersprache. Dass Autoren natürlichsprachlicher Texte ähnliche Funktionen nicht zur Verfügung haben, ist umso erstaunlicher, als es in den vergangenen Jahrzehnten verschiedene Forschungsprojekte und Überlegungen gab, Unterstützung für das Schreiben am Computer unter Berücksichtigung der jeweils verwendeten Sprache zu entwickeln.

Berücksichtigt man, dass Texte heute hauptsächlich am Computer verfasst werden und also elektronisch vorliegen, bietet sich für das Finden von Fehlern in natürlichsprachlichen Texten der Einsatz von Orthographie- und Grammatikprüfprogrammen an. Jedoch werden fast alle der von uns in Anhang A zusammengetragenen Fehler von solchen Programmen nicht erkannt [siehe Gubelmann 2010], und die Qualität der Ergebnisse ist nicht sehr überzeugend, wie wir in Abschnitt 3.3.2 zeigen.

In dieser Arbeit gehen wir daher der Frage nach, wie sich solche typischen Fehler *vermeiden* lassen. Hauptsächlich interessiert uns, ob es möglich ist, Autoren von natürlichsprachlichen Texten ähnlich spezifische Werkzeuge anzubieten, wie sie Programmierer in ihren Editoren finden. Dazu werden wir uns vor allem mit dem Redigieren als Teil des Schreibprozesses und mit der Gestaltung von Schreibwerkzeugen beschäftigen. Die ursprüngliche Motivation liegt in einem Text von Richard Hamlet, in dem er eine Schreibumgebung vorschlägt, die sich an den Möglichkeiten orientiert, die Editoren für Programmiersprachen bieten, und Strukturen der natürlichen Sprache verwendet, um regelbasierte

3. Wir unterscheiden *Operationen* als Abstraktion für alle Änderungen, die ein Autor an einem Text vornimmt, von *Funktionen* in Editoren, die vom Benutzer aufgerufen werden, um solche Operationen umzusetzen.

Funktionen zur Bearbeitung des Texts zu implementieren. [Hamlet 1986] Seine Idee wurde jedoch nie umgesetzt.[4]

Zunächst gehen wir in Abschnitt 1.1 näher auf das Schreibwerkzeug – Textbearbeitungsprogramme – und dessen Charakteristika ein. In Abschnitt 1.2 stellen wir die Hypothesen vor, die dieser Arbeit zugrundeliegen. In Abschnitt 1.3 machen wir den wissenschaftlichen Beitrag der vorliegenden Arbeit deutlich. Anschliessend erläutern wir in Abschnitt 1.4 die von Donald A. Norman [1981] vorgeschlagene Klassifizierung von Fehlern und zeigen deren Übertragung auf Fehler in Texten. Abschnitt 1.5 gibt einen Überblick über den Aufbau der Arbeit.

1.1 Schreibwerkzeuge – Begriffsklärung

Für Computerprogramme, mit denen Texte geschrieben und bearbeitet werden können, existieren verschiedene Begriffe. In der Literatur finden wir *Textverarbeitung*, *Editor*, *word processor*, *word processing system* oder *free-form text editor*. Daneben finden wir die Bezeichnungen *text editor* und *program editor* [vgl. van Dam und Rice 1971, S. 93].

Text, der mit einem *Texteditor* (engl. *text editor*) bearbeitet wird, ist dabei nicht Text im linguistischen Sinne, also «[…] un ensemble d'idées rassemblées sur un thème, structuré et cohérent ayant un début et un fin» [Zock 1985, S. 4], sondern wird im informatischen Sinn verstanden. Der Begriff *Texteditor* bezeichnet Editoren, die die Bearbeitung von Text als *plain text* ermöglichen, entsprechend der *MIME*-Definition:

> Plain text does not provide for or allow formatting commands, font attribute specifications, processing instructions, interpretation directives, or content markup. Plain text is seen simply as a linear sequence of characters, possibly interrupted by line breaks or page breaks. [Freed und Borenstein 1996, S. 6]

Texteditoren werden für Texte verwendet, für die während der Erstellung und Bearbeitung der Inhalt und nicht die Form im Vordergrund steht. So verwenden Programmierer Texteditoren sowohl zum Schreiben von Programmen – also von Text in einer formalen Sprache – als auch zum Erstellen von Dokumentationen – also für natürlichsprachlichen Text, der Elemente verschiedener formaler Sprachen enthält, etwa mathematische Formeln oder Code-Beispiele. Technische Dokumente sind klar strukturiert und entlang dieser Dokumentenstrukturen kann man navigieren, Informationen anfordern oder diese Strukturen manipulieren. Einige Texteditoren bieten die Möglichkeit, den Cursor über Tastenkombinationen entlang bestimmter Strukturen wie Klammerausdrücken, Blöcken oder Funktionsdefinitionen zu bewegen. Diese Funktionalität ist schon in frühen Editoren wie Z [Wood 1981] in den 1970er Jahren vorhanden.

Werden Inhalt und Form klar voneinander getrennt wie beim Schreiben von Texten unter Verwendung von Auszeichnungssprachen wie *HTML*, *troff* oder

4. Persönliche Kommunikation mit Richard Hamlet, E-Mail vom 19. Juni 2006, Message-ID `<200606192000.k5JK01Lu020806@adara.cs.pdx.edu>`.

LaTeX, sind Texteditoren sehr gut geeignet. Bekannte Vertreter sind *vi* oder *XEmacs*. Wir verwenden hierfür den Begriff *Texteditor*.[5]

Textbearbeitungsprogramme (engl. *word processors*) behandeln *Text* im informatischen Sinne als *rich text*. Solche Dateien enthalten neben dem eigentlichen Text im Sinne von *plain text*, der den Inhalt umfasst, immer Angaben zur Formatierung, die bei der Darstellung im entsprechenden Editor ausgeführt werden müssen. [vgl. Freed und Borenstein 1996, S. 3 und S. 6]. Bekannte Vertreter sind heute etwa *OpenOffice* (OO) oder *Microsoft Word* (MS *Word*).

Hausdorf [2005] spricht von *Textproduktionssystemen*. Üblicherweise wird der Begriff *Textverarbeitungsprogramm* verwendet. Wir bevorzugen den deutschen Begriff *Textbearbeitung*, um Konfusionen mit eher computerlinguistisch ausgerichteten Aktivitäten zur tatsächlichen *Verarbeitung* von Text zu vermeiden. Beim Schreiben geht es um die Erstellung und *Bearbeitung* von Text, nicht um die Verarbeitung von fertigen Texten und darin enthaltener Information.

Texteditoren und Textbearbeitungsprogramme unterscheiden sich hauptsächlich in den Funktionen, die einem Autor angeboten werden, im verwendeten Dateiformat und in der Darstellung der Texte; «But the key distinction is more cultural than technical: text editors are used by programmers to write programms and edit system files; word processors are used by everyone to do everything else.» [Haigh 2006, S. 14] Für einen Vergleich von frühen Texteditoren und Textbearbeitungsprogrammen verweisen wir auf den Beitrag von Meyrowitz und van Dam [1982b].

Textbearbeitungsprogramme werden heute hauptsächlich mit der Erstellung und Bearbeitung von Text im linguistischen Sinne assoziiert, der in einer natürlichen Sprache geschrieben ist.

> The term [word processing – C. M.], created on the model of "data processing," is more vague than commonly believed. A human editor, for example, obviously processes words, but is not what is meant by a "word processor." A number of software programs processes words in one way or another—a concordance or indexing program, for example—but are not understood to be word processing programs. [Eisenberg 1992, S. 268]

1.2 Hypothesen

In dieser Arbeit geht es um Unterstützung für Autoren beim Redigieren natürlichsprachlicher Texte. Die Zielgruppe sind dabei erfahrene Autoren, d. h. Personen, die gewohnt sind, Texte zu verfassen, und die gewohnt sind, Texte am Computer zu schreiben. Wir schliessen hier Personen ein, die professionell schreiben (also in einer bezahlten Arbeit, etwa als Journalist), die das Verfassen von Texten als Teil ihrer Arbeit auffassen (etwa Wissenschaftler) und die solche Texte vor der Veröffentlichung redigieren (Korrektoren und Lektoren). *Schreiben* bezieht hierbei alle Aktivitäten von der Planung über das Erstellen einer ersten Textfassung bis zum Erstellen einer endgültigen Fassung ein. Wir

5. Software wie *Eclipse* ist hingegen eine *Programmierumgebung*.

beschäftigen uns mit dem Schreiben von Texten in deutscher Sprache. Wir betrachten Text im linguistischen Sinne ohne Einbezug von Eigenschaften, die eher zum Begriff Dokument gehören – etwa genretypisches Layout, Papiergrösse, Typographie, Verwendung von Grafiken, Tabellen etc.

Heutige Textbearbeitungswerkzeuge bieten Funktionen zum Schreiben und Redigieren von natürlichsprachlichen Texten: Dazu gehören das Einfügen und Löschen von einzelnen Zeichen oder Zeichenfolgen sowie Funktionen wie Ausschneiden und Einfügen (`cut & paste`) oder Suchen und Ersetzen (`search & replace`), die mit markierten Zeichenfolgen arbeiten. Diese Funktionen gehören zu den Kernfunktionen von Textbearbeitungsprogrammen und sind seit den ersten Entwicklungen in diesen Werkzeugen enthalten. Autoren von natürlichsprachlichen Texten stehen jedoch keine Funktionen zur Verfügung, die auf linguistischen Einheiten operieren, also auf Wörtern, Wortgruppen, Satzgliedern oder Sätzen, und deren Eigenschaften berücksichtigen. Es können zwar Orthographie- und Grammatikprüfungen durchgeführt werden, diese sind jedoch keine Editierfunktionen im eigentlichen Sinn, sondern eher im Bereich Nachbearbeitung von Texten zu verorten.

Spezielle Editoren für Programmierer bieten Funktionen, die den Kernfunktionen von Textbearbeitungsprogrammen entsprechen. Zusätzlich bieten solche Editoren jedoch spezielle Funktionen in Abhängigkeit der verwendeten Programmiersprache. Am bekanntesten ist das sogenannte *Syntaxhighlighting*: Einzelne Zeilen des Programms werden entsprechend den syntaktischen Strukturen eingerückt, spezifische Schlüsselwörter werden farbig hervorgehoben. Zudem kann auf den syntaktischen Strukturen navigiert werden, Struktureinheiten können manipuliert, d. h. editiert, werden. Diese Funktionen werden als sprachbewusst oder sprachbasiert (engl. *language-aware* oder *language-based*) bezeichnet, siehe [Ballance et al. 1992, Morris und Schwartz 1981, Van De Vanter 1995, Van De Vanter und Boshernitsan 2000].

Natürlichsprachlicher Text ist nicht nur eine Abfolge von Buchstaben, Leer- und Interpunktionszeichen, sondern vielmehr die Anordnung von Wörtern, Wortgruppen, Satzgliedern und Sätzen entsprechend den Intentionen des Autors unter Berücksichtigung der morphologischen und syntaktischen Regeln einer Sprache: «Coherent texts are not just simple sequences of clauses and sentences, but rather complex artifacts that have highly elaborate rhetorical structure.» [Marcu 2000, S. 395] Natürlichsprachlicher Text weist also wie Programmcode strukturelle Elemente auf, die als Grundlage für Funktionen verwendet werden können, die denen für die Arbeit mit Programmiersprachen ähneln.

Daiute und Taylor [1981] definieren bereits 1981 die Funktionen von Textbearbeitungsprogrammen als «…the ability to create text and carry out various editing functions on it: inserting, deleting, moving, and altering characters, words, phrases, sentences, paragraphs, etc.» [Daiute und Taylor 1981, S. 85], sie beziehen sich also auf linguistische Einheiten. Solche Funktionen finden wir jedoch weder in damaligen noch in heutigen Programmen.

Entsprechend den morphologischen und syntaktischen Regeln einer Sprache können linguistische Strukturen eines Textes ermittelt und kategorisiert werden. Diese Strukturen und Elemente können dann in Funktionen verwendet werden, die sprachbasierten Funktionen in Texteditoren entsprechen. Wir be-

zeichnen sprachbasierte oder sprachbewusste Funktionen für das Schreiben und Redigieren natürlichsprachlicher Texte als *linguistisch unterstützte Editierfunktionen*.

In dieser Arbeit innerhalb des Projektes *LingURed* (**Ling**uistisch **U**nterstütztes **Red**igieren) gehen wir zwei Hypothesen nach:

H1: Analog zu sprachbasierten Funktionen in Editoren für Programmiersprachen ist es möglich, sprachbasierte Funktionen für Textbearbeitungsprogramme zu implementieren, die auf den strukturellen Einheiten der verwendeten natürlichen Sprache operieren.

H2: Linguistisch unterstützte Funktionen können unter Verwendung einfacher computerlinguistischer Ressourcen implementiert werden.

Mit *LingURed* entwerfen wir ein Konzept, das die Erkenntnisse der Schreibforschung bezüglich des Schreib- und Redigierprozesses berücksichtigt und gezielt computerlinguistische Ressourcen einbezieht, um linguistisch motivierte Editierfunktionen zu implementieren. Diese Editierfunktionen sind dem Ziel verpflichtet, typische Redigierfehler zu *vermeiden*.

Wir integrieren diese Funktionen in einen bereits existierenden Editor, um Autoren ihre gewohnte Arbeitsumgebung zu garantieren. Erfahrene Schreiber, die im Umgang mit «ihrem» Textbearbeitungswerkzeug geübt sind, sind nicht bereit, für einige zusätzliche neue Funktionen zu einem ganz neuen Werkzeug zu wechseln. Dies lässt sich durch Evaluationen von Projekten im Bereich Schreibunterstützung und durch die Erkenntnisse der Schreibforschung belegen, wie wir in Kapitel 3 und Abschnitt 5.2 zeigen. Zusätzliche Funktionen werden daher in bereits existierende Programme integriert, es besteht keine Notwendigkeit, ganz neue Werkzeuge zu erstellen.

Die genannten Hypothesen stützen sich darauf, dass die verwendeten computerlinguistischen Ressourcen in einer ausreichenden Qualität vorliegen. Wie wir in Abschnitt 3.2 zeigen, werden heute verfügbare Programme und Funktionen, die computerlinguistische Methoden und Ressourcen verwenden, bislang lediglich für die *Nachbearbeitung* von Texten und die *Fehlerkorrektur* eingesetzt. Aus dem in Kapitel 2 dargestellten Stand der Forschung zum Schreibprozess lässt sich jedoch ableiten, dass ein Bedarf an interaktiver Unterstützung *während* des Redigierens und Editierens besteht. Entsprechende Funktionen müssen also interaktiv benutzt werden können. Der Stand der Technik hinsichtlich Rechengeschwindigkeit und Speicherkapazität und der allgemeine Stand der Forschung bezüglich computerlinguistischer Ressourcen im Bereich Morphologie und Syntax lassen vermuten, dass es möglich sein sollte, interaktive Funktionen zu implementieren, die computerlinguistische Ressourcen verwenden. Die Bearbeitung der präsentierten Hypothesen geht daher von drei Annahmen aus:

A1: Heutige Hardware ist so leistungsfähig, dass rechenintensive Prozesse (etwa die Bestimmung von linguistischen Einheiten und deren morphosyntaktischen Eigenschaften mittels computerlinguistischer Ressourcen) so schnell ausgeführt werden können, dass sie in interaktive Funktionen integrierbar sind.

A2: Die Schreibforschung untersucht unter anderem, wie Autoren mit Schreibwerkzeugen umgehen. Es gibt Computerprogramme, die protokollieren, welche Aktionen ein Autor ausführt, während er einen Text schreibt und redigiert. Diese Programme werden in der aktuellen Schreibforschung sowohl in Laborsituationen als auch in alltäglichen Schreibsituationen eingesetzt. Die Erkenntnisse können wir verwenden, um zu bestimmen, welche Redigieroperationen inhärent fehleranfällig sind und also von einer automatischen Unterstützung profitieren.

A3: Die benötigten computerlinguistischen Ressourcen für Deutsch sind (frei) verfügbar, sie können in andere Programme integriert werden. Die Ergebnisse werden in einer Form geliefert, die die Weiterverarbeitung ermöglicht. Ihre Qualität ist geeignet, um in praxistauglichen Anwendungen (engl. *real-world applications*) (d. h. ohne Beschränkungen auf eine Domäne, ein Genre oder eine Benutzergruppe) eingesetzt zu werden.

Wir bewegen uns mit dieser Arbeit also im Bereich *Sprachtechnologie*, wie ihn Heid [2010] als Integration von Sprachverarbeitungssystemen und Erkenntnissen aus der Verarbeitung grosser Mengen von Sprachdaten definiert zur «[…] Entwicklung praxisrelevanter Verfahren, für den Einsatz bei Experten oder Endkunden». [Heid 2010, S. 361]

Annahme 1 kann ohne Weiteres bestätigt werden: Die Steigerung der Leistungsfähigkeit heutiger Hardware wird in einschlägigen Zahlen der IT-Branche sichtbar. Standardlaptops sind sehr viel kleiner und leichter als noch vor zehn Jahren, sie haben leistungsfähigere Prozessoren und mehr Arbeitsspeicher. Die Entwicklung verläuft weiterhin entsprechend *Moore's Law* [Moore 1965, 1975].

Anwendungen im Bereich Computerlinguistik stellen besondere Anforderungen; wir zeigen daher für die von uns in Betracht gezogenen und schliesslich ausgewählten Ressourcen jeweils auch die Hardware-Anforderungen und die Geschwindigkeit der Ausführung einzelner Prozesse auf und vergleichen sie (sofern vorhanden) mit entsprechenden Angaben aus weiter zurückliegenden Jahren. Generell gilt, dass Prozesse heute sehr viel schneller ausgeführt werden können als vor zehn Jahren – prinzipiell sind computerlinguistische Systeme heute also aus rein informatischer Sicht für den interaktiven Gebrauch geeignet. Schwitter et al. [2000] argumentieren noch, dass computerlinguistische Methoden zu rechenintensiv seien, um sie in praxistauglichen Anwendungen einsetzen zu können. Dies ist heute nicht mehr der Fall.

Für Annahme 2 zeigen wir, welche der Erkenntnisse der Schreibforschung für uns verwendbar sind, und machen Forschungslücken deutlich.

Für Annahme 3 evaluieren wir verschiedene computerlinguistische Systeme im Bereich Morphologie und automatischer Wortartenbestimmung hinsichtlich ihrer Eignung für den Einsatz in praxistauglichen Anwendungen.

1.3 Wissenschaftlicher Beitrag diese Arbeit

Die vorliegende Arbeit leistet wissenschaftliche Beiträge auf verschiedenen Gebieten:

- Wir wenden den Begriff der *slips* und das Klassifikationsschema von Norman [1981] erstmals auf Fehler in natürlichsprachlichen Texten an und zeigen, dass sich die von Norman [1983] vorgeschlagenen allgemeinen Designprinzipien von Prozessen und Bedienoberflächen zur Vermeidung von *slips* auf Editierfunktionen übertragen lassen. (Abschnitt 1.4) Wir präsentieren eine umfangreiche Sammlung von Fehlern in Texten, die sich als *slips* kategorisieren lassen. (Anhang A)

- Wir übertragen das Prinzip der sprachbasierten Funktionen, wie sie in Programmiereditoren seit langem üblich sind, erstmals auf Funktionen in Textbearbeitungsprogrammen. (Abschnitt 4.2)

- Wir schlagen eine Klassifikation von sprachbewussten Funktionen in Textbearbeitungsprogrammen vor, die nicht nur die beobachtbaren Veränderungen an der Textoberfläche ex post beschreibt, sondern auch die Absicht des Autors und die benötigten Ressourcen zu deren Umsetzung berücksichtigt. Unsere Taxonomie kann zur Spezifizierung von Funktionen in Editoren verwendet werden. (Abschnitt 4.3)

- Wir werten aktuelle Forschungsergebnisse der Schreibforschung aus, die sich mit der Protokollierung von Schreibsitzungen von Autoren beschäftigt. Wir zeigen in dieser Arbeit, dass die Rolle des Schreibwerkzeugs momentan nicht ausreichend berücksichtigt wird. Wir machen deutlich, dass Daten der Schreibforschung unter Gesichtspunkten ausgewertet werden müssen, die über die Betrachtung der Veränderungen des Schreibprodukts hinausgehen, um einen Beitrag zur Erleichterung und Verbesserung des Schreibens zu leisten. (Abschnitt 5.3)

- Wir evaluieren aktuelle Systeme zur morphologischen Analyse und Generierung des Deutschen. (Abschnitt 6.2) Eine solche Evaluation unter Verwendung aktueller Texte wurde seit 1995 (d. h., seit den ersten Morpholympics [Hausser 1996]) nicht mehr durchgeführt. Damit gehen wir über die weit verbreitete Annahme «[…] that tools for linguistic preprocessing and automatic syntactic analysis are widely available nowadays […]» [Evert 2005, S. 20] hinaus. Vielfach werden solche Ressourcen unreflektiert und im blinden Vertrauen auf eine «genügende» Qualität der Ergebnisse verwendet und in Applikationen integriert.Wir zeigen die tatsächlichen Eigenschaften von Ressourcen für Deutsch und berücksichtigen diese in der Implementierung von linguistisch unterstützten Editierfunktionen. (Kapitel 6)

- Wir zeigen die exemplarische Implementierung von linguistisch motivierten Funktionen unter Verwendung geeigneter computerlinguistischer Ressourcen und erörtern, welche Möglichkeiten sich daraus für Autoren ergeben und welche Grenzen berücksichtigt werden müssen. (Kapitel 7)

1.4 Slips

Norman [1981, 1983] schlägt eine Klassifizierung von Fehlern vor, die sich in verschiedenen Bereichen des alltäglichen Lebens beobachten lassen. Er unterscheidet *mistakes* und *slips*:

> Call the highest level specification of a desired action an **intention** […] An error in the intention is called a **mistake**. An error in carrying out the intention is called a **slip**. [Norman 1983, S. 254][6]

Übertragen wir dies auf linguistische Phänomene, sind *mistakes* also im Bereich der Kompetenz, *slips* im Bereich der Performanz nach Chomsky [1965] anzusiedeln. Für *slips* gibt Norman Erklärungen, die die Grundlage seines Klassifizierungsschemas bilden: Die Gründe für fehlgeschlagene Handlungen oder nicht erfolgreich ausgeführte Absichten liegen in der Regel im Design der verwendeten Werkzeuge [Norman 1983].

Wenn wir typische Fehler in deutschen Texten (wie in Anhang A zusammengestellt) analysieren, können wir unter Berücksichtigung der in Textbearbeitungsprogrammen verfügbaren Funktionen Erklärungen finden, die den von Norman vorgeschlagenen Klassen für *slips* entsprechen. Norman [1983] bezieht diese Klassen auf Prozesse allgemein, um eine Erklärung für objektiv beobachtbare Fehler zu finden. Er schlägt Prinzipien zum Design von Bedienoberflächen vor, um *slips* zu vermeiden. Textbearbeitungsprogramme sind eine spezielle Art von Computerprogrammen und sollten ebenfalls diesen allgemeinen Prinzipien folgen.

Die folgenden der von Norman postulierten Klassen von *slips* können für typische Redigierfehler in Texten verwendet werden, die mit Textbearbeitungsprogrammen erstellt wurden. Wir geben jeweils ursprünglich angeführte Beispiele und Beispiele für das Editieren von Text an[7]:

- «Forgetting an intention (but continuing with the action sequence)»

  allgemein Sich an einem Ort befinden, ohne zu wissen, was man dort will und wie man dorthin gelangt ist [Norman 1981, S. 9].

  editieren Während des Navigierens an eine Textstelle vergisst der Autor, was er dort ändern oder kontrollieren wollte.

- «Capture error (When a sequence being performed is similar to another more frequent or better learned sequence, the latter may capture control)»

  allgemein Der Weg von der Arbeit nach Hause und der Weg von der Arbeit zum Fischladen haben einen Teil gemeinsam – auf dem Weg zum Fischladen wird dann jedoch der normale Weg nach Hause weitergefahren und das Einkaufen des Fisches wird vergessen [Norman 1981, S. 5].

  editieren Die Abfolgen der Funktionen für das Verschieben eines Elements und das Kopieren eines Elements sind sehr ähnlich. In jedem Fall wird das betroffene Element zunächst markiert, dann wird für das Verschieben die Funktion `cut` aufgerufen, für das Kopieren jedoch `copy`. Anschliessend sind die Funktionsfolgen wieder identisch: Der Benutzer navigiert an den Zielort im Text und ruft dann die Funktion `paste` auf. Diese Ähnlichkeit kann dazu führen, dass

6. Hervorhebungen im Original.
7. Wir verwenden jeweils die Originalformulierung für die Bezeichnung einer Klasse entsprechend [Norman 1983, S. 255].

etwas, das nur kopiert werden soll, am Ursprungsort fälschlich gelöscht wird, oder etwas, das tatsächlich verschoben werden soll, nur kopiert wird und also am Ursprungsort weiterhin vorhanden ist, wie in diesem Beispiel:

«*Noch gibt es keinerlei Vermutungen, wer in den neu entdeckten Gräbern bestattet wurde. In zwei Wochen werden die archäologischen Arbeiten an den Gräbern gestoppt. Dann werden Umbauarbeiten beginnen. In zwei Wochen werden die archäologischen Arbeiten an den Gräbern gestoppt. Dann werden umfassende Umbauarbeiten beginnen. Die Bischofskirche wird bis 2014 für rund 30 Millionen Euro saniert.*» (Beispiel A.94)

- «Misordering the components of an action sequence (including skipping steps and repeating steps)»

  allgemein Beim Ausnehmen und Putzen von Fisch direkt nach dem Angeln den Fisch über Bord werfen und den Abfall behalten [Norman 1983, S. 255].

  allgemein Vergessen, Wasser in die Kaffeemaschine zu füllen, bevor sie angestellt wird [Norman 1981, S. 10].

  allgemein Erneutes Starten des Motors, obwohl das Auto bereits fährt [Norman 1981, S. 10].

  editieren Einfügen von Text, der bereits vorhanden ist.

  «*Die Entwickler von von* RUSKIN *stellten erst beim Ende des Projektes feststellten, dass die Autoren ganz andere Funktionen gewünscht und gebraucht hätten.*» (Beispiel A.103)

- «False triggering: A properly activated schema is triggered at an inappropriate time»

  allgemein Die Brille absetzen wollen, obwohl man sie nicht aufgesetzt hat [Norman 1981, S. 7].

  allgemein Etwas in ein Diktiergerät sprechen, dies unterbrechen und bei Wiederaufnahme des Diktats statt in das Diktiergerät in den Telefonhörer sprechen [Norman 1981, S. 7].

  editieren Etwas an einer Textstelle aus der Zwischenablage einfügen, obwohl der gewünschte einzufügende Text noch nicht kopiert oder ausgeschnitten wurde.

- «Spoonerism (Reversal of event components)»

  allgemein Vertauschen von Wörtern oder Wortteilen wie *You have tasted the whole worm* statt korrekt *You have wasted the whole term* [Norman 1981, S. 10].

  editieren Vertauschen von Wörtern.

  «*Von der grauhaarigen Frau sieht Christin nur zuckenden den Rücken.*» (Beispiel A.166)

- «Blends (Combinations of components from two competing schemas)»

  allgemein Verschmelzung von *close* und *shut* zu *clut* oder *Rockefeller Brothers Foundation* zu *Rockebrothers* [Norman 1981, S. 10].

editieren Verschmelzungen

«George Washington heißt bei den Haudenosaunee seitdem Hanadahguyus, Stadzerstörerenn sie einen Brief an den Präsidenten schreiben, dann lautet die Anrede stets: Dear Hanadahguyus!» (Beispiel A.211)

- «Triggering of schemas meant only to be thought, not to govern action»

allgemein Schon etwas aussprechen, auf das erst später eingegangen werden soll – man ist in Gedanken schon weiter voraus [Norman 1981, S. 10].

editieren Bereits das nächste Wort im Satz schreiben, bevor das aktuell syntaktisch notwendige geschrieben ist – es fehlt also ein Wort im Satz. Ähnlich ist das Einfügen von Wörtern an einer Textstelle, die noch nicht den richtigen Anschluss an diese Wörter enthält.

«Die Terrorismus und Spionage zuständige Ermittlungsbehörde prüft, ob der Anfangsverdacht einer geheimdienstlichen Straftat vorliegt» (Beispiel A.47)

- «Failure in triggering: When an active schema never gets invoked»

allgemein Man wollte Person A bitten, Kaffee zu kochen, und wundert sich dann, dass kein Kaffee da ist – die Bitte wurde nie ausgesprochen [Norman 1981, S. 11].

editieren Hierunter fallen unvollständige Änderungen – ein Wort soll durch ein anderes ersetzt werden, das ursprüngliche wird jedoch nicht gelöscht. Beim Korrekturlesen fällt dies nicht auf, da nur das neue Wort wahrgenommen wird. Es kann auch vorkommen, dass ein Wort, dass an eine andere Stelle verschoben werden soll, an der ursprünglichen Textstelle ausgeschnitten wird, dann wird jedoch vergessen, dieses Wort an der Zieltextstelle tatsächlich einzufügen.

«In Abschnitt 2.1.2.1 sind wir auf Taxonomien von Redigieroperationen eingegangen, die mehrheitlich aus der Beobachtung von Änderungen in Texten entwickelt wurden, jedoch nicht die ursprüngliche Redigierabsicht Absicht der Autoren berücksichtigen.» (Beispiel A.99)

Solche Fehler sind allgemein darauf zurückzuführen, dass der Benutzer eines Werkzeugs eine *komplexe Abfolge von Funktionen* oder Arbeitsschritten ausführen muss. Sie können vermieden werden, wenn das Werkzeug erlauben würde, eine bestimmte Absicht in nur wenigen Schritten zu verwirklichen. Redigieraktionen müssen vom Autor mit einer geschickten Kombination von zeichenbasierten Funktionen des Textbearbeitungsprogramms umgesetzt werden, siehe Abschnitt 3.1.2. Diese Übersetzung ist kognitiv anspruchsvoll, das eigentliche Redigierziel gerät schnell aus dem Fokus, während der Ausführung von Funktionsfolgen werden notwendige Schritte «vergessen» oder in einer «falschen» Reihenfolge ausgeführt.

Funktionen, die auf sprachlichen Einheiten arbeiten und die Besonderheiten einer bestimmten Sprache berücksichtigen, wären zwar in sich komplex, für den Benutzer jedoch einfacher zu bedienen. Wenn Funktionen auf angemessenen Objekten operieren, kann die Anzahl der notwendigen Funktionen reduziert werden. Somit verringert sich die Gefahr von *slips*, da die Länge der auszuführenden Folge von Funktionen sich drastisch verkürzt. Zudem bedeuten Funktionen, die auf der gleichen Abstraktionsebene implementiert sind, auf der gewünschte

Redigierziele formuliert werden [siehe auch Severinson Eklundh und Kollberg 1996, S. 184], dass der fehleranfällige Übersetzungsprozess in Abfolgen zeichenbasierter Kernfunktionen entfällt. Damit stehen mehr kognitive Ressourcen für die eigentliche Redigieroperation zur Verfügung. Textbearbeitungsprogramme und deren Bedienoberflächen sollten also die allgemeinen Designforderungen für Software generell angemessen umsetzen. Dies bezieht sich nicht nur auf die Anordnung von Menüs, sondern ebenso auf die eigentliche Arbeitsfläche und die implementierten Funktionen.

1.5 Aufbau der Arbeit

In Kapitel 2 stellen wir die Erkenntnisse der Schreibforschung hinsichtlich des Schreibprozesses vor. Wir gehen auf verschiedene Schreibmodelle ein und konzentrieren uns anschliessend auf das Redigieren. Ausgehend von verschiedenen Definitionen dieses Prozesses erörtern wir die Besonderheiten beim Redigieren am Computer.

In Kapitel 3 zeigen wir den gegenwärtigen Stand der Technik von Textbearbeitungsprogrammen und stellen Projekte zur Schreibunterstützung in verwandten Bereichen vor. Wir konzentrieren uns auf Projekte, die computerlinguistische Methoden und Ressourcen verwenden, um die Nachbearbeitung von Texten und die Fehlerkorrektur zu erleichtern.

In Kapitel 4 stellen wir das Konzept des linguistisch unterstützten Redigierens als Lösungsansatz für die in Abschnitt 1.2 formulierten Fragestellungen vor. Ausgehend von sprachbasierten Funktionen in Texteditoren erörtern wir, welche der zugrundeliegenden Ideen und Prinzipien adaptiert werden können, um Funktionen für das Redigieren natürlichsprachlicher Texte zu implementieren. Ermittelt werden operationalisierbare Handlungsanweisungen zur Lösung von linguistischen Problemen, die typische Redigierarbeiten beinhalten oder nach sich ziehen. Wir entwickeln ein Klassifikationsschema von Redigierfunktionen, das sowohl linguistische, textlinguistische, schreibprozessorientierte als auch technische Aspekte berücksichtigt. Auf dieser Grundlage können Funktionsklassen konzipiert werden, deren Varianten sich als konkrete Funktion implementieren lassen. Die Einordnung von Redigierfunktionen in das Klassifikationsschema dient der Bestimmung der Kosten einer Funktion und ihrer Implementierung hinsichtlich der Komplexität in Bezug auf linguistische und technische Ressourcen. Wir legen dar, welche Möglichkeiten der Einsatz computerlinguistischer Methoden und Ressourcen bietet, und verdeutlichen die Grenzen. Daraus leiten sich Grundsätze für die Implementierung solcher Funktionen wie auch für die Gestaltung entsprechender Bedienoberflächen ab.

In Kapitel 5 stellen wir dar, dass Schreibwerkzeug und Schreibprozess nicht als voneinander unabhängige Elemente betrachtet werden können. Dabei gehen wir insbesondere auf das Schreiben mit dem Computer ein. Anhand der aktuellen Literatur der Schreibforschung sowie aktueller Forschungsprojekte zeigen wir, welche Erkenntnisse unser Konzept des linguistisch unterstützten Redigierens beeinflussen und welche Fragestellungen (noch) nicht bearbeitet wurden.

In Kapitel 6 untersuchen wir, welche der heute verfügbaren computerlinguistischen Ressourcen für Deutsch für unsere Anforderungen tatsächlich verwendbar

sind. Die implementierten Funktionen sind ganz klar abhängig von der Verfügbarkeit und der Qualität entsprechender Komponenten. Ausgehend von unseren Implementierungsleitlinien untersuchen wir Komponenten auf ihre prinzipielle Eignung zur Verwendung in interaktiven praxistauglichen Anwendungen und hinsichtlich der Qualität ihrer Resultate. Die jeweils von uns als am besten geeignet ermittelten Systeme werden wir in Funktionen verwenden. Wir zeigen den Einfluss der Ressourcen auf die Umsetzung des Konzepts und die erwartbare Qualität der so implementierten Funktionen. Notwendige Verbesserungen existierender Werkzeuge und Ressourcen werden im Hinblick auf die interaktive Benutzung innerhalb der Zielsetzung dieser Arbeit definiert.

In Kapitel 7 zeigen wir konkrete Umsetzungen der in Kapitel 4 formulierten Leitlinien und grundsätzlichen Überlegungen. Zunächst stellen wir den von uns verwendeten Editor *XEmacs* vor. Anschliessend wählen wir aus den vorgeschlagenen Funktionen beispielhafte Vertreter aus, beschreiben sie entsprechend dem in Abschnitt 4.3 vorgeschlagenen Klassifkationsschema und beschreiben ihre Funktionsweise sowie die konkrete Implementierung. Ähnliche Funktionen lassen sich durch einfache Veränderungen in der Implementierung (z.B. hinsichtlich bestimmter Parameter) ableiten. Wir zeigen hier die Möglichkeiten und Grenzen unseres Konzepts, die durch die gewählte Sprache (Deutsch) und deren linguistische Eigenschaften, durch die Qualität der verfügbaren computerlinguistischen Ressourcen und den Schreibprozess selbst bedingt sind.

Kapitel 8 ist das Fazit der vorliegenden Arbeit. Wir kommen hier noch einmal auf die in Abschnitt 1.2 formulierten Hypothesen zurück und stellen dar, wie das in Kapitel 4 vorgestellte Konzept des linguistisch unterstützten Redigierens tatsächlich geeignet ist, die Forschungslücke zu schliessen. Unser Fazit stützt sich vor allem auf die in Kapitel 7 exemplarisch ausführlich beschriebenen und implementierten Funktionen. Anschliessend diskutieren wir Konzept und Umsetzung im Hinblick auf die Ausrichtung dieser Arbeit. Ein weiteres Element im Schlusskapitel ist der Ausblick auf Forschungsfragen, die sich im Umfeld des linguistisch unterstützten Redigierens in verschiedenen Forschungsbereichen stellen.

Anhang A beinhaltet eine Sammlung von Redigierfehlern aus verschiedenen realen Texten. Diese Fehler sind nach verschiedenen Gesichtspunkten kategorisiert, dazu wird jeweils eine Erklärung für die Entstehung eines Fehlers gegeben (soweit bekannt oder nachvollziehbar).

2 Redigieren als Element im Schreibprozess

Writing a text is a complex task that needs a coordinated implementation of a large set of mental activities.

—Denis Alamargot & Lucile Chanquoy, *Through the Models of Writing*

Some precious old words do seem to change,
'Cause that's what life's all about: To arrange and rearrange and rearrange.

—Pete Seeger, *Arrange and Rearrange*, 1997

Die Schreibforschung beschäftigt sich mit den *Prozessen*, die bei der Entstehung und Bearbeitung von Texten involviert sind, mit den verwendeten *Medien* und *Werkzeugen* und mit der *Qualität* des produzierten Textes selbst.[1] Text wird hier entsprechend der Formulierung von Zock verstanden: «Un texte peut être défini comme la mise en forme d'un ensemble d'idées rassemblées sur un thème, structuré et cohérent ayant un début et un fin.» [Zock 1985, S. 4]

Literaturwissenschaft, Linguistik und Computerlinguistik beschäftigen sich ebenfalls mit Texten. Sie gehen jedoch davon aus, dass diese Texte nicht mehr verändert werden, also «fertig» sind. Es wird versucht, möglichst viele Informationen aus einem Text oder einer Textmenge zu gewinnen. Die Schreibforschung fokussiert dagegen stark auf den Entstehungs- und Veränderungsprozess

1. Die Konzentration auf den Prozess (also das Schreiben und dessen verschiedene Phasen) statt auf das Produkt (den fertigen Text) erfolgte in der Forschung in den USA und Kanada in den 1960er Jahren, im deutschsprachigen Raum etwa Mitte der 1970er Jahre [Merz-Grötsch 2005, S. 77].

von Text und auf die involvierten Personen, deren mentale Prozesse und die verwendeten Werkzeuge; Aspekte, die in der Literatur- oder Sprachwissenschaft nicht oder nur am Rande berücksichtigt werden. [Vgl. Hawisher et al. 1995, Zock 1985] Gleiches gilt für die Computerlinguistik. Die verschiedenen Sichtweisen auf «Text» unterscheiden sich wesentlich und lassen sich pointiert so ausdrücken: «[…] computational linguistics relies on fixity of text and composition deals with fluidity of text»[2].

Nur ein kleiner Bereich der Computerlinguistik beschäftigt sich mit dem Entstehungsprozess von Text: automatische Textgenerierung (engl. *text generation by computer* oder *natural language generation*). Dabei wird davon ausgegangen, dass die zu kommunizierende Information *strukturiert* und *vollständig* vorliegt. Beide Voraussetzungen sind für das Schreiben in der Regel nicht gegeben, siehe [Zock 1999, 2001, Zock et al. 2004]. Die automatische Generierung von Text ist in Bereichen erfolgreich, in denen strukturierte Daten für den Menschen leserfreundlich aufbereitet und dargestellt werden sollen – etwa Wetterberichte, medizinische Daten oder Börsenmeldungen. Wie Reiter und Dale [2000] zeigen, wird die gegebene Datenstruktur in natürlichsprachlichen Text umgewandelt. Es findet *während* dieses Prozesses jedoch keine Änderung der verwendeten Informationen, der beabsichtigten kommunikativen Ziele oder des Stils statt. Gerade diese Aspekte beeinflussen jedoch das Redigieren von Text durch den Autor während des Schreibens. Textgenerierung ist – anders als Schreiben – nur nach sorgfältiger Planung des zu erstellenden Textes möglich. Diese Textpläne resultieren aus Faktoren, die sich aus den vorhandenen Daten, der Textsorte und dem Kommunikationsziel ergeben und werden anschliessend automatisch umgesetzt. Planen und die Umsetzung des Plans sind strikt voneinander getrennt. Für Autoren und für das Schreiben gelten solche strengen Vorgaben nicht, wie wir anhand der Schreibmodelle in Abschnitt 2.1.1 zeigen. Da wir uns hier auf das Redigieren von Texten durch Autoren konzentrieren, sind die Methoden, wie sie Reiter und Dale [2000] darstellen, nur am Rande relevant.

Schreiben ist ein *Prozess*, der verschiedene Aktivitäten beinhaltet, mit verschiedenen Werkzeugen ausgeführt werden kann und zu einem *Produkt*, dem Text, führt. Dieser Text soll in seiner endgültigen Version unterschiedlichen Ansprüchen genügen, die sich aus den Zielen des Autors sowie aus Vorgaben von Verlagen, Auftraggebern etc. ergeben. Dabei spielen auch immer das Schreibwerkzeug und das Schreibmedium eine Rolle. Das Schreiben mit dem Computer war von Beginn an ein Objekt der Schreibforschung. Eine wichtige Frage war, ob die Verwendung eines «neuen» Werkzeuges Auswirkungen auf die entstehenden Texte, den Schreibprozess selbst oder einzelne Phasen (etwa das Redigieren) hat. Typisch sind Fragen wie: «What do people do when they write? What is writing? How do people go about it? How do we find out about it and model it accurately?» [Williams und Holt 1989, S. X] Und natürlich wurde immer auch versucht zu erfassen, ob diese Auswirkungen positiv oder negativ zu beurteilen sind, siehe etwa [Taylor 1987, S. 79] [Hill et al. 1991, S. 83] [Collier und Werier 1995, S. 48] [Molitor-Lübbert 1997, S. 48] oder [Hartley 2007, S. 293].

2. Scott Warnock in einer Nachricht im Forum «Making Word Processors Process Words» (moderiert von Cerstin Mahlow und Michael Piotrowski) im LMS *SmartSite* (basierend auf *Sakai*) der UC Davis, für den Online-Teil der Konferenz «Computers and Writing 2009», 25.2.2009, 12:28 PST.

Die Antworten darauf sind so vielfältig wie widersprüchlich. Neue Erkenntnisse fliessen in Modelle des Schreibprozesses ein. In Abschnitt 2.1 stellen wir die Entwicklung von Modellen zur Beschreibung des Schreibprozesses vor und gehen in Abschnitt 2.2 auf die kognitiven Anforderungen beim Schreiben ein. In Abschnitt 2.3 widmen wir uns dem Redigieren als einem wesentlichen Teil des Schreibprozesses.

2.1 Schreibprozess und Schreibmodelle

Schreibprozessforschung hat mit methodischen Schwierigkeiten zu kämpfen: Das entstandene endgültige Produkt liefert keine Hinweise auf den Entstehungsprozess. Berücksichtigt man alle Zwischenprodukte vom ersten Entwurf bis zur endgültigen Fassung zu verschiedenen Zeitpunkten, lässt sich die Entwicklung eines Textes nachvollziehen. Beobachten lassen sich auch die Aktionen, die ein Autor ausführt, wie etwa Molitor-Lübbert schildert: «Sichtbar ist bei allen schreibenden WissenschaftlerInnen jedoch, daß sie ihre Zeit darauf verwenden, abwechselnd zu lesen, zu schreiben und nachzudenken.» [Molitor-Lübbert 1997, S. 49] Aus diesen Beobachtungen können mit Modellen, die wir im folgenden Abschnitt 2.1.1 vorstellen, Prozesse und Phasen erklärt und beschrieben werden. In Abschnitt 2.1.2 gehen wir genauer auf die Methoden ein, mit denen solche Modelle erstellt werden.

2.1.1 Schreibmodelle

Flower und Hayes [1981] entwickelten das erste Modell des Schreibprozesses, das Schreiben als Lösen eines rhetorischen Problems sieht [Flower und Hayes 1981, S. 369], wobei sich die verschiedenen Phasen überlappen und auch rekursiv auftreten können, siehe Abbildung 2.1. Dabei könne das *Problem* sowohl ein

Abbildung 2.1: Schreibm von Flower und Hayes [S.

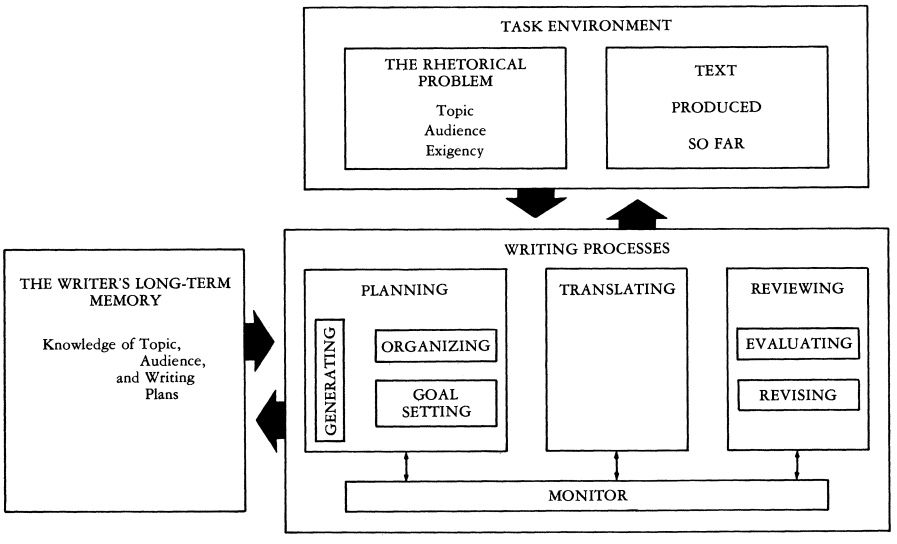


Figure 1. Structure of the writing model. (For an explanation of how to read a process model, please see Footnote 11, pages 386-387.)

Der Text selbst, also das Schreibprodukt, ist in diesem Modell nur in der Rolle als *bislang geschriebener Text* (engl. *text produced so far*) vorhanden. Dieser Text ist ständiger Veränderung unterworfen und so die sichtbare Manifestation der Schreibabsicht des Autors zu einem gegebenen Zeitpunkt. Text wird

hier ebenfalls nur im linguistischen Sinne betrachtet, also unabhängig etwa von typographischen Aspekten. Der *Text* im Sinne des *fertigen* Produktes, des erstellten Dokuments, wird nicht erwähnt.

Figure 2
Hayes's (1996) Framework for Understanding Cognition and Affect in Writing

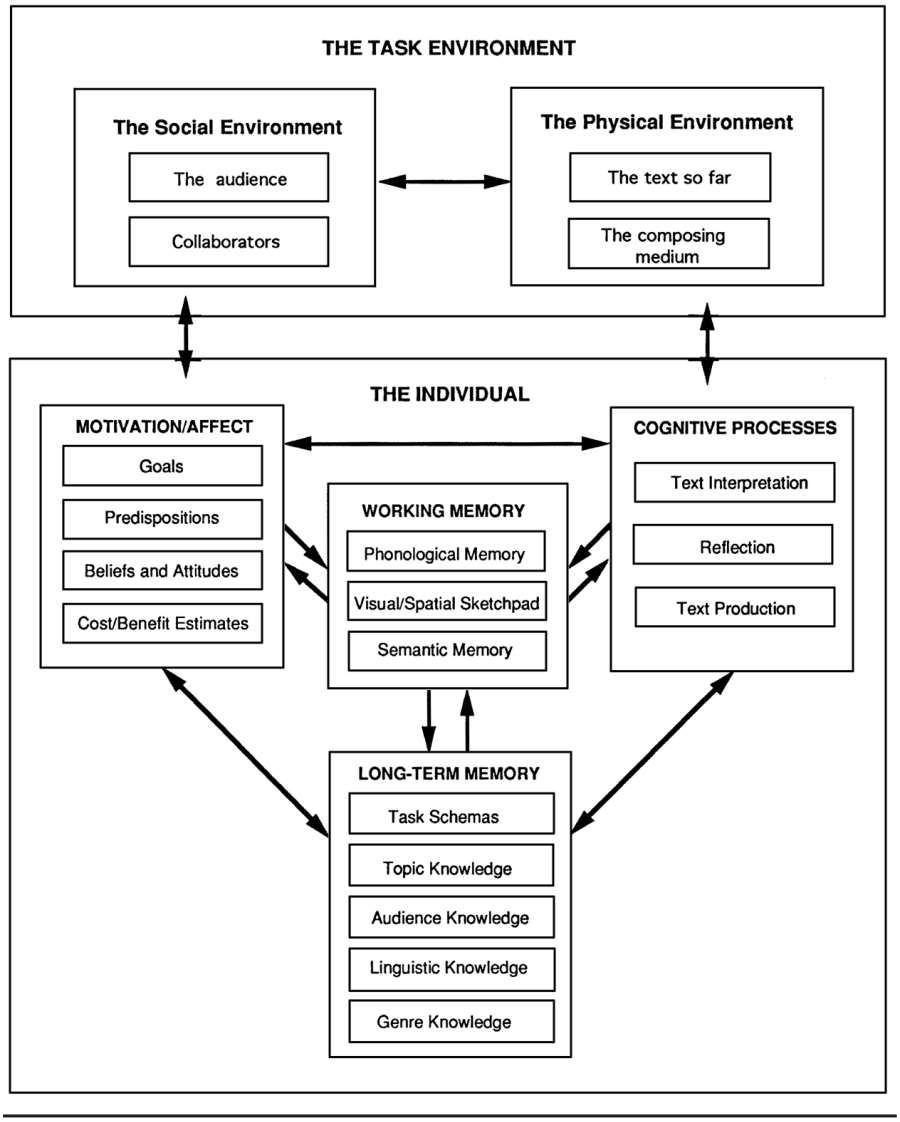


Abbildung 2.2: Aktualisiertes Schreibmodell von Hayes [1996, S. 4].

sion significantly slowed sentence production in this task. They also found that the pacing of the sentence production process was changed. Ordinarily, sentences are produced in bursts. That is, the writer produces a sequence of words in rapid order, then pauses, produces another sequence, pauses, and so

Hayes [1996] modifizierte das ursprüngliche Modell unter stärkerer Berücksichtigung involvierter kognitiver Prozesse und der Beziehungen des Autors zu seinem Umfeld bzw. der Schreibsituation, siehe Abbildung 2.2. Wie der entstehende Text (hier nur noch als *text so far* bezeichnet) gehört hier auch das verwendete Werkzeug, genauer das Medium, zur physischen Umgebung. Text und Werkzeug interagieren sowohl mit der sozialen Umgebung als auch mit dem Individuum.

Das Modell von Kellogg [1996] stellt die Beziehungen zwischen Arbeitsgedächtnis und Informationsverarbeitung in den Vordergrund, siehe Abbildung 2.3 auf der nächsten Seite. Bereiter und Scardamalia [1987] fokussieren ebenfalls auf die mentalen Prozesse und unterscheiden *Knowledge Telling* und *Knowledge Transforming*, siehe Abbildung 2.4 auf der nächsten Seite. In beiden Modellen werden weder der geschriebene Text noch das verwendete Werkzeug berücksichtigt.

2.4.2 Kellogg's (1996) model

The objective of Kellogg (1996) is to integrate, in a unique model, writing processes and a system of information processing (*Cf.* Figure 7).

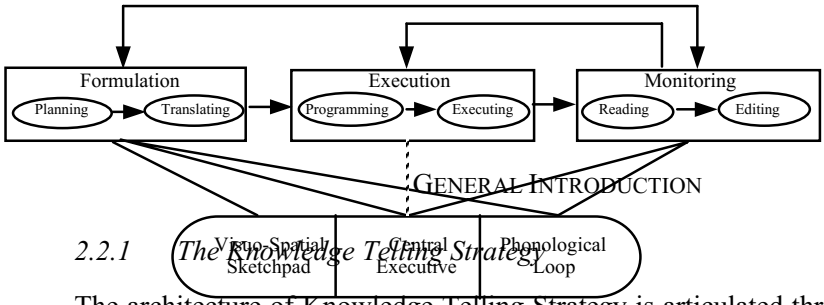


Abbildung 2.3: Modell der Schreibprozesse von Kellogg [1996, S. 58].

2.2.1 The Knowledge Telling Strategy

9

GENERAL INTRODUCTION

The architecture of Knowledge Telling Strategy is articulated through three co nents (*Cf.* Figure 2).

Figure 7: Relationships between production and Working Memory components,adapted from Kellogg (1996). Copyright © 1996 by Lawrence Erlbaum Associates. Adapted with permission.

The system is the model of Working Memory such as developed by Baddeley (1986). The Working Memory architecture is composed of a Central Executive globally dedicated to complex processes (*i.e.* reasoning, reflecting, *etc.*) and two slave registers, the Visuo-Spatial Sketchpad and the Articulatory Loop, allowing to maintain visual and phonological representations.

Concerning the architecture of writing processes, Kellogg (1996) resumes the writing model elaborated by Brown, McDonald, Brown and Carr (1988). He distinguishes three components – the Formulation, the Execution and the Monitoring – each comprising two basis processes.

- 'Formulation' is composed of two processes, the Planning and the Translating. The first process allows to fix goals that have to be reached, to research relevant domain knowledge units, according to these goals, and to organise the retrieved information. The second one allows the transformation of ideas into linguistic structures.
- 'Execution' allows the motoric elaboration (Programming) and the realisation of the message (Executing). The output of the Formulation component is then programmed before its transcription. The system of handwriting or typing consists of graphically reproducing the message. The programming and the muscular executing vary as a function of the final mode of emission.
- 'Monitoring' implies to read the message and can lead to a possible new editing of this message. The first process (Reading) allows to regularly reread/verify the message, during its elaboration. The second process, Editing, com-

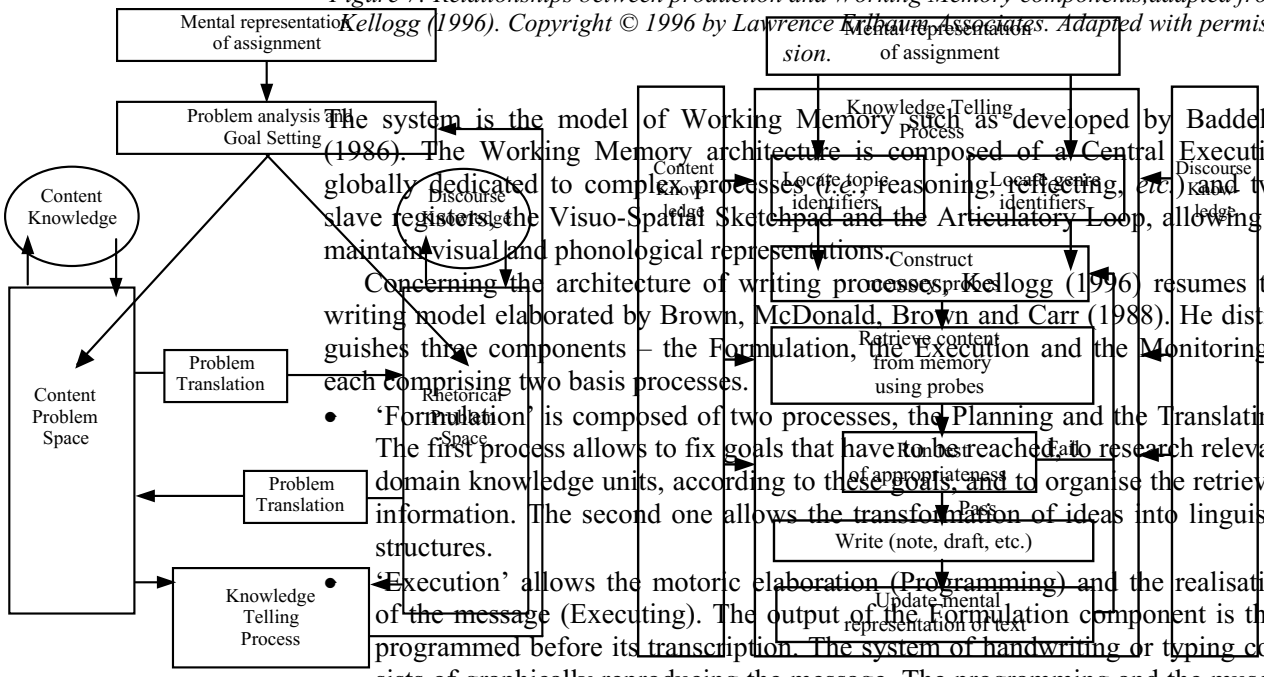


Abbildung 2.4: Knowledge Transforming (links) und Knowledge Telling (rechts).

Figure 3: Knowledge Transforming Strategy,adapted from Bereiter & Scardamalia (1987). Copyright © 1987 by Lawrence Erlbaum Associates. Adapted with permission.

The main difference between the two strategies is the system of information processing. Transforming Strategy of complex problem-solving is more elaborate than the Knowledge Telling Strategy. Mental Representation of Assignment component allows the expert writer to elaborate a complex process. The interactive functioning of three specific components of the Knowledge Transforming Strategy supports this problem-solving system.

- The 'Problem analysis and Goal Setting' component allows the expert writer to realise, according to writing instructions, a complex analysis of (a) the task to be carried out, (b) the objectives to reach, and (c) the necessary means to reach this objective. This planning activity on the task allows to control two 'sub-problems' solving, respectively linked to content planning ('what to tell') and to the rhetorical process planning to express these contents ('how to tell').
- These two kinds of planning are managed in their own frames. Those of the Problem Space', related with the Content Knowledge, transforms or modifies the domain knowledge. The 'Rhetorical Problem Space' is related with the two other components, Discourse Knowledge, retrieves or modifies pragmatic knowledge. The 'Translating' process, A central point in the functioning principle of the Knowledge Transforming

Sharples (1999) sieht Schreiben als Design und betont den kreativen Aspekt. Sein Modell ist ein Kreislauf der verschiedenen Phasen ohne explizites Anfang- und Endkriterium. Siehe Abbildung 2.5 auf der nächsten Seite. Hoc

Heute ist sich die Forschungsgemeinschaft einig, dass die verfügbaren Modelle nicht ausreichen, um das Schreiben in seiner Komplexität zu beschreiben. [Wengelin et al. 2010, S. 730]

Dies betrifft sowohl nicht direkt beobachtbare mentale Prozesse als auch tatsächlich beobachtbare Prozesse. Die Entwicklung von Schreibprozessmodellen ist weiter Gegenstand der Forschung. Solche Modelle sollen auf Erkenntnissen aus Studien beruhen, die verschiedene Aspekte des Schreibprozesses genauer

Hayes

Bis auf das modifizierte Modell von Hayes, berücksichtigt keines der Modelle die Rolle des Schreibwerkzeugs, obwohl Schreiben ohne Werkzeug und Medium

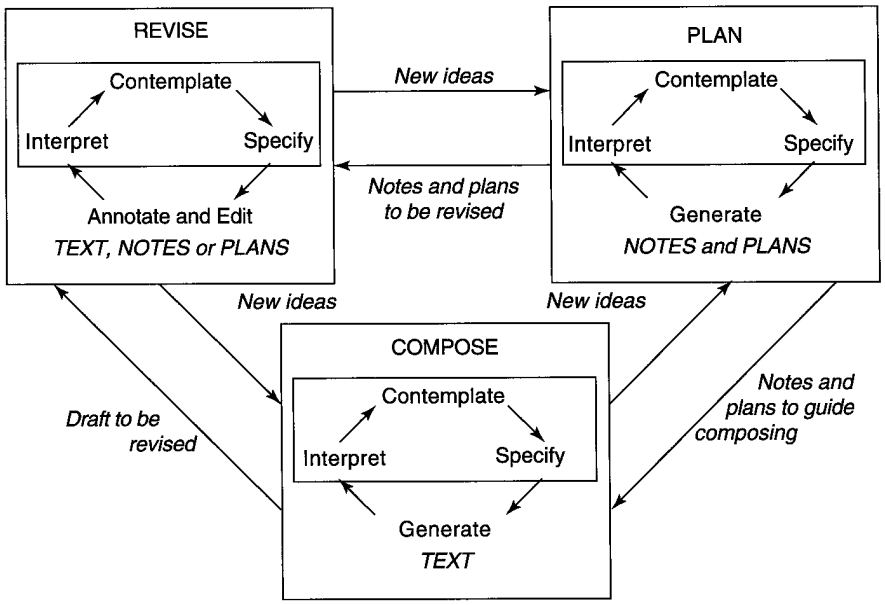


Abbildung 2.5: Modell der Schreibprozesse von Sharples [1999, S. 72].

nicht möglich ist. Werkzeuge beeinflussen aufgrund ihrer verfügbaren Funktionen und ihres Designs die Handlungen, die mit ihnen ausgeführt werden. Norman [1988, S. 9ff] hat dies als *affordances* eingeführt, als die möglichen (intendierten und nichtintendierten) Beziehungen zwischen Objekt und Benutzer aufgrund der Eigenschaften des Objekts.

Ein grosses Manko ist zudem die Tatsache, dass die Modelle lediglich dazu genutzt werden können, beobachtete Prozesse *ex post* zu erklären. Es ist (noch) nicht möglich, diese Modelle zur *Vorhersage* von Prozessen oder Aktionen zu verwenden [Alamargot und Chanquoy 2001, S. 23 und S. 28]. Ziel ist die Entwicklung von Modellen mit einer ähnlichen Aussagekraft, wie sie andere kognitive Modelle besitzen: «In our field, models have not yes achieved the level of explicitness often found in cognitive psychology. Indeed, some psychological models are explicit enough to be embodied in runable computer programs that can be used to identify the models predictions.» [Hayes 2001, S. 234]

Im Bereich der automatischen Textgenerierung finden wir dagegen Modelle, die Teile des Schreibprozesses beschreiben – vor allem das Planen auf Diskursebene – und tatsächlich implementiert sind. Andriessen et al. [1996] und Reiter und Dale [2000] präsentieren einen guten Überblick. Allerdings wird auch hier festgestellt, dass die Situation insgesamt unbefriedigend sei: «Computational research is gradually moving from the descriptive to the procedural phase. That is still a long way from modelling strategies.» [Andriessen et al. 1996, S. 272]

2.1.2 Methoden der Schreibforschung

Flower und Hayes [1981] stützten sich bei der Entwicklung ihres Schreibmodells auf Erkenntnisse, die sie mit *lautem Denken* (engl. *think aloud protocols*) gewannen. Die Probanden sollten während des Schreibens kommentieren, was sie tun und warum sie es tun. Flower und Hayes wandten diese Methode, die aus dem Bereich des Problemlösens bekannt war, als erste auf die Beobachtung von Schreibprozessen an: «The main method used was protocol analysis,

which requires writers to verbalize their thoughts as they write and to produce additional comments and notes later.» [Holt 1989, S. 53]

Die Probanden sind beim lauten Denken mit zwei Aufgaben beschäftigt: dem Schreiben selbst und dem Kommentieren aller Aktionen. Untersuchungen zu solchen *dual tasks*[3] zeigen jedoch, dass die Anforderung, eine weitere Aufgabe neben dem Schreiben auszuführen, einen Einfluss sowohl auf den Schreibprozess selbst als auch auf das Produkt hat:

> […], when participants wrote while performing a visuo-spatial secondary task, they paused for longer durations even after text production had begun, and maintained this pattern throughout the remainder of the writing period.
>
> Finally, the quality of the essays, determined by trained judges blind as to the testing conditions, showed pervasive effects when a secondary task was introduced. [Levy und Ransdell 2002, S. 26f]

Damit ist fraglich, inwieweit die Erkenntnisse, die aus solchen Experimenten mit lautem Denken gewonnen werden, tatsächlich den normalen Schreibprozess beschreiben: «Although the validity of the verbal protocol method is more valid than directed retrospection, it is also potentially more disruptive of working memory resources needed for writing.» [Olive et al. 2002, S. 32][4] Solche zusätzlichen Anforderungen, wie auch akustische irrelevante Einflüsse, führen zu einer unnatürlichen Schreibsituation und beeinträchtigen so den Schreibprozess, den sie eigentlich untersuchen möchten, wie Janssen et al. [1996] zeigen (siehe auch [Faigley und Witte 1981, S. 412] und [Chenoweth und Hayes 2003, S. 116]).

Auch Holt [1989] kritisiert die Methode des lauten Denkens. Es sei unüblich, beim Schreiben zu sprechen, daher führe diese Anforderung zu einer unüblichen und keiner normalen Schreibsituation. Zusätzlich beinhalte Schreiben sehr viele unbewusste Prozesse. Es sei daher fraglich, ob die aus lautem Denken gewonnenen Schreibmodelle tatsächlich verwendbar sind [siehe Holt 1989, S. 50f]. Offensichtlich liegt hierin also die beklagte Unzulänglichkeit der Schreibmodelle begründet.

Es stehen jedoch auch andere Möglichkeiten zur Verfügung. Bereits Lutz [1983] wählt zu Beginn der 1980er Jahre Methoden zur Aufzeichnung von Daten während des Redigierens mit dem Computer und auf dem Papier, die auf Verbalisierung verzichten, denn: «Despite their many defenders, talking-out-loud protocols are unnatural, and writers may end up critiquing their cognitive processes rather than simply reporting on them.»[Lutz 1983, S. 160] Ihr Versuchsaufbau sieht vor, beim Redigieren am Computer alle fünf Sekunden den aktuelle Zustand des Textes zu speichern, inklusive einer Kennzeichnung der Zeilen, an denen seit der letzten Speicherung etwas geändert wurde, und dazu, was dort geändert wurde. Beim Redigieren auf Papier müssen die Versuchspersonen Stifte in verschiedenen Farben für jeden Durchlauf verwenden und

3. «The dual task paradigm is so-named because people are asked to perform a primary task (such as composing a text) while simultaneously engaged in a secondary (or loading) task that is portrayed as having less importance.» [Levy und Ransdell 2002, S. 10].
4. Bei der gelenkten Retrospektion (engl. *directed retrospection*) [Ericsson und Simon 1980] werden die Probanden gefilmt und kommentieren ausgewählte Szenen nach der Aufnahme.

die Änderungen nummerieren. Eine ausführliche Beschreibung findet sich in [Lutz 1983, S. 156f]. Damit erhält der Versuchsleiter für beide Varianten ein Protokoll, das es erlaubt, die Arbeit am Text nachzuvollziehen. Lutz verwendet diese Daten, um eine Taxonomie von Redigieroperationen zu erstellen, siehe Abschnitt 2.3.3. Sie berücksichtigt jedoch explizit nur die Veränderung der Textoberfläche, die mentalen Prozesse können nicht einbezogen werden.

An dem ursprünglichen Schreibmodell von Flower und Hayes richten sich bis heute Schreibanleitungen aus und unterscheiden *vorbereiten* (lesen, exzerpieren, Zielpublikum bestimmen, Schreibplan und Gliederung entwerfen), eine *Rohfassung schreiben* und *überarbeiten* (auf zwei Ebenen: inhaltlich sowie sprachlich und formal), etwa von Kruse et al. [2006, S. 15f]. Diese Phasen lassen sich nicht klar voneinander abgrenzen, sie treten im Schreibprozess mehrfach auf [Andriessen et al. 1996, S. 253]. Die Länge einzelner Phasen sowie Sequenzen von Abfolgen unterscheiden sich jeweils sowohl entsprechend dem Stands des Schreibprojektes als auch zwischen verschiedenen Autoren. Schreiben ist kein linearer, sondern ein rekursiver Prozess, verschiedene Subprozesse interagieren, wechseln sich ab und unterbrechen sich gegenseitig. [vgl. Collier und Werier 1995, Hill et al. 1991, Kellogg 1988, 2001, Kollberg und Severinson Eklundh 2002, Kruse et al. 2006, Olive et al. 2002, Severinson Eklundh 1994, Sharples 1994]

Auch wenn die Methoden zur Ermittlung der ablaufenden Prozesse beim Schreiben also nicht optimal sind und kein exaktes Bild vermitteln können, lässt sich aus den so gewonnenen Daten ableiten und durch *dual tasks* oder sogar *triple tasks* [Olive et al. 2002] belegen, dass Schreiben hohe kognitive Anforderungen stellt [Torrance und Galbraith 2006]. Im folgenden Abschnitt 2.2 gehen wir genauer auf diese Anforderungen ein.

2.2 Kognitive Anforderungen beim Schreiben

Zahlreiche Studien betonen, dass Schreiben hohe kognitive Anforderungen stellt. Kellogg [1999] vergleicht die Anforderungen gar mit jenen des Schachspielens. Die involvierten Teilprozesse müssen vom Autor koordiniert werden. Durch Übung können einige der Prozesse, wie beispielsweise korrekte Rechtschreibung oder Zeichensetzung, fast «automatisiert» und simultan mit anderen ausgeführt werden [vgl. Fayol 1999, Kellogg 2008]. Die menschliche kognitive Kapazität ist jedoch durch die verfügbaren Ressourcen[5] naturgemäss begrenzt. Die kognitiven Ressourcen müssen auf die verschiedenen Aufgaben, die beim Schreiben gelöst werden sollen, aufgeteilt werden, wie Allen und Scerbo [1983], McCutchen [1996], Quinlan et al. [2009], Ransdell und Levy [1996, 1999], Torrance und Galbraith [2006] und Van Waes et al. [2010] zeigen. Die konkrete Gestaltung des Schreibprozesses und der Subprozesse ist zudem abhängig vom Schreiber und den von ihm bevorzugten Schreibstrategien oder Schreibstilen, die zusätzlich durch die gewählte (oder vorgegebene) Sprache, das Thema, das Zielpublikum, das Genre etc. beeinflusst werden.

5. «[…] *resources* refer to the mental energy available at a given moment that cognitive processes require to operate […] *Capacity* corresponds to the maximal amount of resources that is available to an individual. […] *cognitive effort* corresponds to the amount of resources required by a given task […]» [Piolat et al. 2004, S. 21] Hervorhebung im Original.

Allein das nächste zu schreibende Wort auszuwählen und aufzuschreiben, stellt hohe Anforderungen, da Bedingungen auf verschiedenen Ebenen berücksichtigt werden müssen:

> Deciding, which word to use next, for example, is constrained, at least, (a) by the syntax of the sentence to which it contributes, (b) by the writer's intended message, by the need to maintain coherence across sentences, (c) by whether a direct or anaphoric reference is appropriate, (d) by the perceived register of the discourse community to whom the text is directed, and (e) by the need to avoid overusing the same word. The processing associated with managing each of these constraints is likely to be complex and to make some demand on processing resources. [Torrance und Jeffery 1999, S. 5][6]

Wenn wir das Schreiben eines Textes mit allen involvierten Vorgängen gesamthaft betrachten, lassen sich verschiedene Aufgaben definieren, die kognitive Kapazität beanspruchen: die Bestimmung der Funktion eines Textes unter Berücksichtigung des Zielpublikums, die Beachtung des Themas des Textes, die Bearbeitung bereits geschriebenen Textes (engl. *text produced so far (TPSF)*) unter Berücksichtigung von Kohärenz und Wortwahl, um möglichst ohne Redundanzen einen kohäsiven Text zu erzeugen, siehe [Fayol 1999, S. 13] und [Alamargot und Chanquoy 2001, S. 1].

Hinzu kommen noch eher physische Prozesse, wie das Schreiben mit einem Stift – was nur ein langsames Manifestieren von Textelementen erlaubt – oder die Fähigkeit, Text am Computer effizient einzugeben – wer Tastaturschreiben gelernt hat, kann Text wesentlich schneller eingeben als jemand, der nur beide Zeigefinger oder gar nur einen[7] verwendet. Die kognitiven Anforderungen werden also durch die Wahl des Schreibwerkzeugs unter Umständen erhöht, wie Dowling beobachtet:

> Effects attributed to this problem include conscious selection of short words and simple sentence structures, a general tendency towards summary in preference to a fuller exposition of ideas, and (c) [sic!] conscious rejection of certain character combinations found to be awkward, particularly those involving the outer two fingers. [Dowling 1994, S. 229][8]

Taylor [1987, S. 79] spricht in diesem Zusammenhang von der «brain-to-hand-to-keyboard-to-screen-connection». Einen Einfluss hat auch die Benutzung der Maus, die für das Aufrufen einiger Funktionen notwendig sein kann. Dies erhöht die kognitiven Belastungen noch, wie Dowling [1994, S. 230] in Interviews mit erfahrenen Schreibern zeigen konnte.

---

6. Eigentlich sollte "by the need to maintain …" besser einen eigenen Eintrag erhalten.
7. Es gibt tatsächlich Personen, die einen Zeigefinger zur Bedienung aller Buchstaben- und Zifferntasten verwenden, der andere Zeigefinger bedient nur die Shift-Taste. Durch Übung kann selbst diese Art der Eingabe fast «blind» erfolgen.
8. Offenbar sollte die Aufzählung durch Einfügung von Markierungen explizit gemacht werden, es ist jedoch nur die Markierung «*(c)*» erhalten.

Wir gehen in Abschnitt 5.3 noch genauer auf die Zusammenhänge zwischen Schreibwerkzeug und Schreibprozess ein. Nachdem wir uns bislang mit dem Schreibprozess allgemein beschäftigt haben, konzentrieren wir uns im folgenden Abschnitt auf einen Subprozess, das Redigieren.

2.3 Redigieren

Der Begriff des Schreibprozesses lässt sich pragmatisch als die Gesamtheit aller Prozesse und Aktionen definieren, die prinzipiell notwendig sind oder von einem Autor tatsächlich ausgeführt werden, um einen Text zu produzieren. In der Regel wird *Text* dabei im Sinne des resultierenden Dokumentes verstanden, also inklusive Arbeitsschritten zur Formatierung u. ä. Die Abgrenzung der Subprozesse voneinander bzw. die Definition der einzelnen Subprozesse, wie sie in den verschiedenen Schreibmodellen enthalten sind, gestaltet sich dagegen wesentlich schwieriger.

Wir beschäftigen uns hier genauer mit dem Prozess des Überarbeitens oder *Redigierens*. Im folgenden Abschnitt 2.3.1 beschäftigen wir uns mit verschiedenen Sichtweisen und Definitionen des Begriffs Redigieren. Anschliessend gehen wir im Abschnitt 2.3.2 genauer auf das Redigieren am Computer ein, berücksichtigen also den Einfluss des Werkzeugs auf den Prozess. In Abschnitt 2.3.3 stellen wir vor, nach welchen Gesichtspunkten Redigieroperationen klassifiziert werden können, und gehen in Abschnitt 2.3.4 auf die Aufzeichnung von Redigieroperationen ein, die die Grundlage für die vorher vorgestellten Klassifizierungen sind.

2.3.1 Begriffsklärung und Definition

Zunächst geht es um eine Begriffsklärung. In englischsprachiger Literatur ist von *post-writing*, *reviewing*, *revising* und *editing* die Rede. Wir fassen *revising* und *editing* als die Äquivalente zum deutschen Begriff *Redigieren* auf. Zur Veranschaulichung verwenden wir zwei Texte, die mit Redigieranweisungen versehen wurden.

Abbildung 2.6 auf der nächsten Seite zeigt ein Redemanuskript[9] von US-Präsident Barack Obama. Abbildung 2.7 auf der nächsten Seite zeigt eine Detailaufnahme. Dieses Foto hat innerhalb der Schreibforschergemeinschaft der USA einige Kommentare und Diskussionen ausgelöst:

> The volume of Obama's editing is unusual but not unheard of. The quality of his editing is exceptional for a public figure. Think of just one sentence in the shot above. The original says "This has always been our history." Obama changes it to, "This has always been the history of our progress." A different, more interesting, and more original-sounding thought.[10]

9. Es ist eine Rede vor der Sitzung beider Kammern des Kongresses im September 2009 im Zusammenhang mit der Gesundheitsreform der USA.
10. James Fallows, «About That Extraordinary Photo of an Obama-edited Speech», 25.3.2010, 7:42 PM ET, online http://www.theatlantic.com/politics/archive/2010/03/about-that-extraordinary-photo-of-an-obama-edited-speech/38065/ (zuletzt besucht am 8.12.2010, 19:34).

Abbildung 2.6: US-Präsident Barack Obama mit dem Manuskript einer Rede.[11]

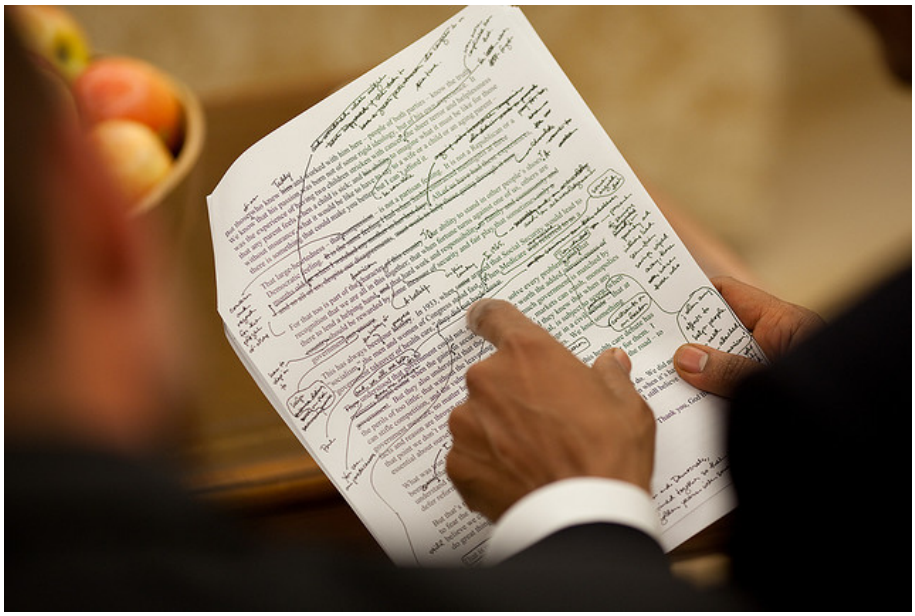
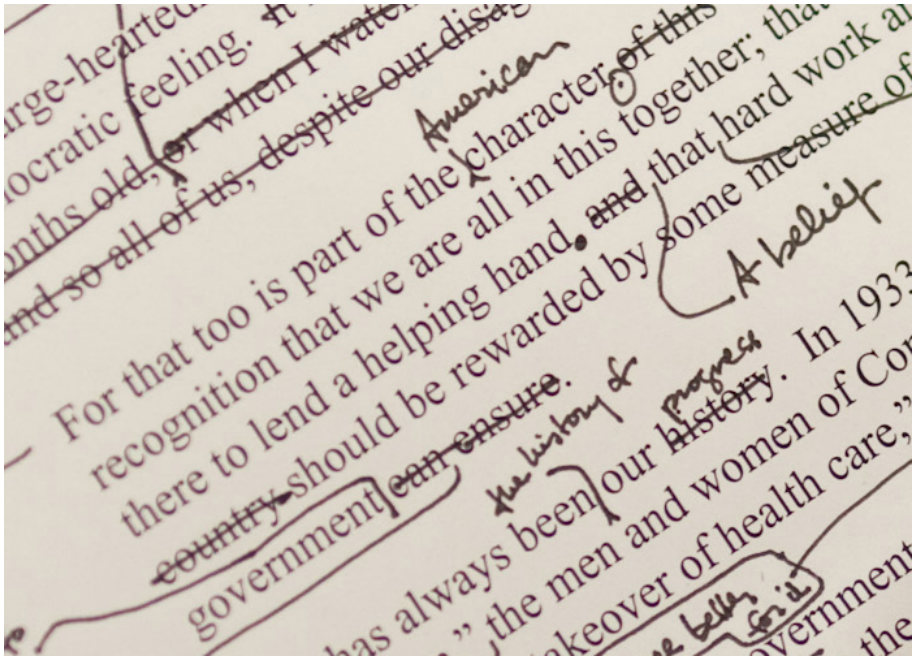


Abbildung 2.7: Detailaufnahme des redigierten Manuskripts von Barack Obama.[12]



Ein anderer kommentiert:

> What did Obama do to the draft? Repeatedly, he added agency, attributing acts, feelings and thoughts to specific people or groups of people. He added nouns and pronouns and active verbs, converting verbal nouns in the original draft to active verbs with human predicates (Teddy, our seniors, members of Congress, I, we, etc.). He drew individuals, members of Congress generally, and

11. http://www.flickr.com/photos/whitehouse/4455914253/sizes/l/in/set-72157623676571910/ (zuletzt besucht am 8.12.2010, 19:34), United States Government Work.
12. http://www.theatlantic.com/politics/archive/2010/03/about-that-extraordinary-photo-of-an-obama-edited-speech/38065/ (zuletzt besucht am 8.12.2010, 19:34).

> the American people as active agents into a joint national project of compassion and positive action.[13]

Die Änderungen, die Obama vornimmt, sind also Redigieroperationen, die die gewünschte Aussage stärker betonen, einen anderen Fokus setzen, die Bedeutung ändern. Er ersetzt einzelne Wörter, fügt Wortgruppen hinzu, entfernt Satzglieder, trennt lange Sätze in kürzere Sätze auf etc. Die Kommentatoren verwenden zur Beschreibung der vorgenommenen Änderungen explizit linguistische Bezeichnungen («*active verbs*», «*pronouns*» etc.).

Abbildung 2.8 zeigt eine redigierte Manuskriptseite von Max Frisch, die auf der Schreibmaschine erstellt wurde. Auch hier wird markiert, dass Wörter eingefügt, Satzglieder gestrichen, zunächst ergänzte Sätze vollständig entfernt werden sollen. Für beide Texte können ähnliche Redigieroperationen festgestellt werden, die sich im Textbild manifestieren.

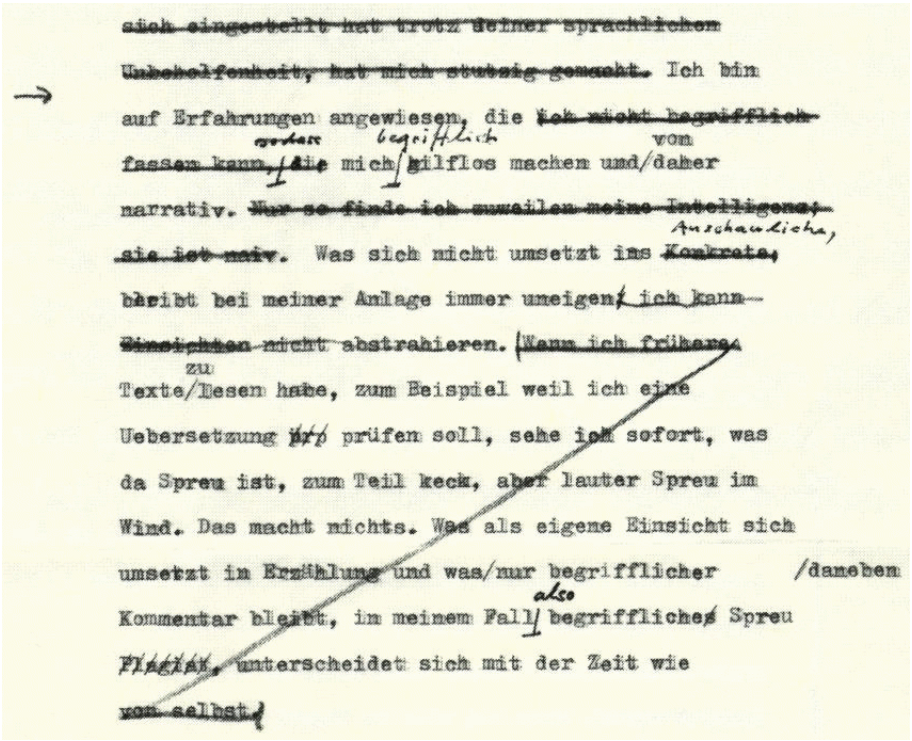

sich eingebettet hat trotz deiner sprachlichen
Unbeholfenheit, hat mich stubsig gemacht. Ich bin
auf Erfahrungen angewiesen, die ich nicht begrifflich
sondern begrifflich vom
fassen kann, die mich hilflos machen und daher
narrativ. Nur so finde ich zuweilen meine Intelligenz,
Anschauliche
sie ist naiv. Was sich nicht umsetzt ins Konkrete,
bleibt bei meiner Anlage immer uneigen, ich kann
Einzelnen nicht abstrahieren. Wenn ich früher
zu
Texte/lesen habe, zum Beispiel weil ich eine
Uebersetzung prüfen soll, sehe ich sofort, was
da Spreu ist, zum Teil keck, aber lauter Spreu im
Wind. Das macht nichts. Was als eigene Einsicht sich
umsetzt in Erzählung und was/nur begrifflicher /daneben
also
Kommentar bleibt, in meinem Fall/ begriffliches Spreu
Fiktion, unterscheidet sich mit der Zeit wie
von selbst.

Abbildung 2.8: Redigierte Manuskriptseite aus «Entwürfe zu einem dritten Tagebuch» *von Max Frisch [Frisch 2010, S. 178].*

In beiden Fällen erfolgten diese Revisionen höchstwahrscheinlich in einer expliziten Phase: Obama erhielt die «fertige» Rede und modifizierte sie, bevor er sie hielt, Frisch markierte Änderungen, die in einer nächsten Fassung des Manuskripts vorgenomen werden sollten. Dies scheint die bis in die 1980er Jahre vertretene Annahme zu bestätigen, dass der Schreibprozess eher linear ist. An dessen Ende gibt es eine explizite Phase des Überarbeitens und Korrigierens: das Redigieren.

Diese Auffassung änderte sich jedoch und «[…] the revision process was redefined *as a sequence of changes in a composition—changes which are initiated by cues and occur continually throughout the writing of a work.*» [Sommers 1980, S. 380][14] Ähnlich definiert auch Lutz diesen Prozess: «Though given various

13. Andrew Sprung, 26.3.2010, «A MORE PERFECT SPEECH DRAFT: OBAMA EDITS "OUR" NATIONAL STORY», online http://xpostfactoid.blogspot.com/2010/03/more-perfect-speech-draft-obama-edits.html (zuletzt besucht am 8.12.2010, 19:34).
14. Hervorhebung im Original.

Hayes and Flower in their 1980 model. In parallel with these theoretical viewpoints about revision, other models or theoretical ideas have been elaborated. These are presented in the following paragraphs.

2.2 Other specific models of revision

2.2.1 Scardamalia and Bereiter's (1983, 1985) model

Soon after Hayes and Flower's (1980) general model about the writing process, Scardamalia and Bereiter (1983, 1985) proposed a more complex and complete description of the revising activity. Before describing the authors' perspective, we would like to specify that their proposal cannot be considered as a real model of revision, but rather, in an educational view, as a technique to help writers to revise. Nevertheless, psychologists have used this architecture to understand the complexity of the revising process.

According to Scardamalia and Bereiter, the functioning of revision would work through a self-regulated procedure, composed of three recursive mental operations that would chain cyclically during a stage of revision. These operations are respectively labelled by the authors as 'Compare', 'Diagnose' and 'Operate', and the general procedure is called 'C.D.O. procedure' (*Cf.* Figure 17).

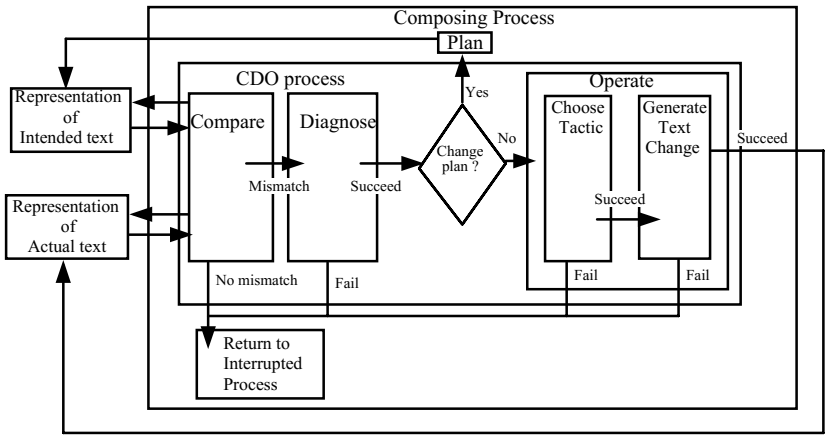


Abbildung 2.9: Modell des Redigierprozesses von Scardamalia und Bereiter [1983].

Figure 17: Model of the CDO process, adapted from Scardamalia and Bereiter (1983). Copyright ©1983 by John Wiley & Sons. Adapted with permission.

Auch Fitzgerald [1987] verwendet in ihrer Definition diese Idee, formuliert jedoch allgemeiner:

> Revision means making any changes at any point in the writing process. It involves identifying discrepancies between intended and instantiated text, deciding what could or should be changed in the text and how to make desired changes, and operating, that is, making the desired changes. [Fitzgerald 1987, S. 484]

Auf diese Definition bezieht sich Hayes [2004] und stellt fest, dass Redigieroperationen nicht nur durch das Auftreten und Erkennen von *Fehlern* ausgelöst würden, sondern durch *Erkenntnisse*, die beim Schreiben oder beim Lesen des Geschriebenen gewonnen würden, und so die Überarbeitung eines Textteils notwendig machten [Hayes 2004, S. 10f]. Dieser Aspekt des Redigierens erscheint ihm sogar wesentlich interessanter; er konstatiert:

> However, revisions that are stimulated by the discovery of new connections, new ideas or new arguments seem intrinsically more interesting. Such revisions are likely to be associated with improvements in the substance rather than the form of the text. [Hayes 2004, S. 20]

Zock [1998] hält fest, dass während des Redigierens Inhalte und rhetorische Ziele noch nicht vollständig bestimmt sind. Die zentralen Ideen ergeben sich

erst bei der Strukturierung der verwendeten Fakten, die wiederum von diesen abhängig sei. Die rhetorischen Ziele beeinflussten Inhalt und Struktur eines Textes. Zock folgert daher: «Il y a donc interaction mutuelle entre les idées, la structure de texte et les effets rhétoriques (buts). […] l'effet rhétorique est souvent un effet de bord d'un assemblage particulier des données.» [Zock 1998, S. 245]

Wir finden hier auch einen Bezug zur Kreativität, wie wir ihn in Kapitel 1 für das Schreiben als gesamten Prozess beschrieben haben: «[…] Evaluation itself can lead to the discovery of new possibilities, not just errors.» [Flower et al. 1986, S. 25]. Noch expliziter bringt es Breidenbach [2006] zum Ausdruck, die Redigieren als kreativen Akt des Ausprobierens und Entdeckens betrachtet: «To be creative, however, revision needs time and freedom from excessive constraint and regimentation. It needs to remain open and loose and walk on the edge of possibilities, trying them on and checking them out.» [Breidenbach 2006, S. 200]

Wie die Untersuchungen von Wengelin et al. [2009] zeigen, wird nur in etwa 20 % der Fälle, in denen bereits Geschriebenes noch einmal gelesen wird, tatsächlich ein Fehler gefunden und korrigiert – mehrheitlich geht es um Redigieren, das mit dem eigentlichen Schreiben eng verwoben ist und nicht in einer expliziten Redigierphase stattfindet. Hayes und Chenoweth [2006] berichten von etwa 10 % «verschwendeten» (*wasted*) Tastendrücken, die Fehler seien und korrigiert werden müssten. Berücksichtigt man den gesamten Schreibprozess, also auch explizite Redigierphasen nach dem Abschluss einer Zwischenfassung, sind mindestens ein Drittel aller vorgenommenen Änderungen eindeutig *keine* Korrekturen von orthographischen oder grammatikalischen Fehlern [siehe auch Faigley und Witte 1981].

Leijten et al. [2010a], Olive et al. [2009] und Van Waes et al. [2010] zeigen, dass Autoren bereits *vor* dem Beenden des aktuellen Satzes redigieren. Sharples [1999, S. 105ff] unterscheidet *revising while composing* von *revising as seperate activity*. Alle bestätigen also die Definition von Fitzgerald, dass Redigieren zu jedem beliebigen Zeitpunkt stattfinden könne. Diese Beobachtungen gelten unabhängig vom verwendeten Schreibwerkzeug, treffen also auf das Schreiben und Redigieren mit Papier und Stift, mit der Schreibmaschine wie auch mit Textbearbeitungsprogrammen zu.

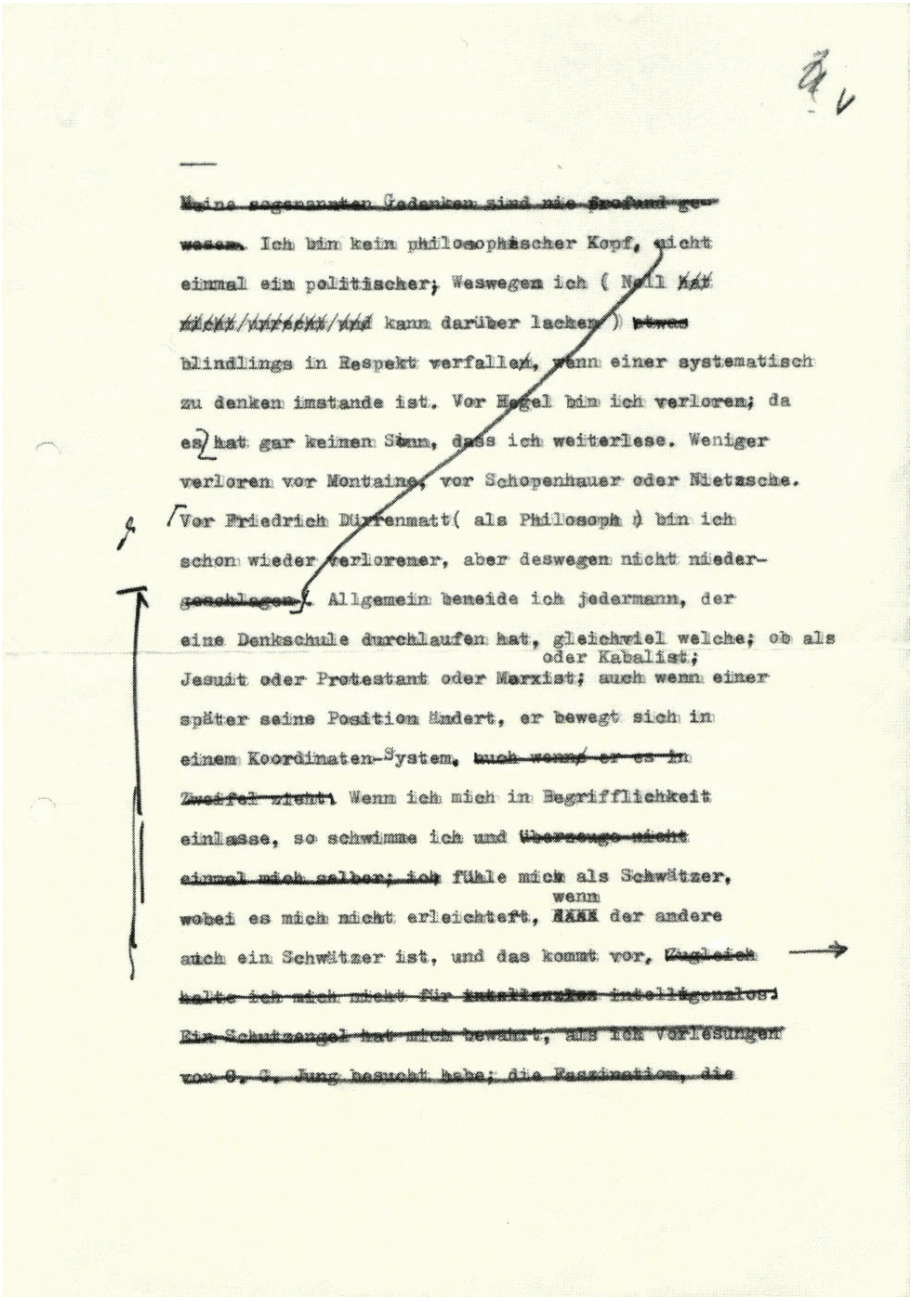
Autoren planen nur kleinere Strukturen (Wörter und Wortgruppen) voraus [Chanquoy et al. 1990, Olive et al. 2009], die dann geschrieben werden. Dies bestätigt allgemeine Aussagen zur Kapazität des Arbeitsgedächtnisses: «[…] we can process no more than two to four elements at any given time with the actual number being at the lower than the higher end of this scale.» [Sweller 1999, S. 4] Menschen können also weniger als die von Miller [1956] genannten «magischen sieben» Elemente verarbeiten.

Die verschiedenen Definitionen von Redigieren beziehen sich auf den Prozess ohne Berücksichtigung des verwendeten Werkzeugs. Wir können feststellen, dass Redigieren einen kreativen Aspekt enthält. Jede Änderung zu jedem beliebigen Zeitpunkt im geschriebenen wie auch im noch zu schreibenden Text wird als Redigieren bezeichnet. Darin ist eingeschlossen, dass nicht erst vollständig geschriebene Sätze geändert werden. Vor allem wird nur ein kleiner

Teil des Redigierens als Finden, Diagnostizieren und Korrigieren von Fehlern bezüglich expliziter Regeln verstanden.

2.3.2 Redigieren am Computer

Autoren redigieren Texte, unabhängig davon, mit welchem Werkzeug sie arbeiten. Abbildung 2.10 zeigt eine weitere Manuskriptseite von Max Frisch. Dieser mit der Schreibmaschine erstellte Text wurde, vermutlich bereits während des Schreibens, mit den Mitteln der Schreibmaschine redigiert – beispielsweise die Streichungen in der vierten Zeile oder das Ersetzen eines Wortes in der fünftletzten Zeile.



Ich bin kein philosophischer Kopf, nicht
einmal ein politischer; Weswegen ich (Nell
kann darüber lachen))
blindlings in Respekt verfalle, wenn einer systematisch
zu denken imstande ist. Vor Hegel bin ich verloren; da
es hat gar keinen Sinn, dass ich weiterlese. Weniger
verloren vor Montaigne, vor Schopenhauer oder Nietzsche.
Vor Friedrich Dürrenmatt(als Philosoph) bin ich
schon wieder verloremer, aber deswegen nicht niederAllgemein beneide ich jedermann, der
eine Denkschule durchlaufen hat, gleichviel welche; ob als
oder Kabalist;
Jesuit oder Protestant oder Marxist; auch wenn einer
später seine Position ändert, er bewegt sich in
einem Koordinaten-System,
Wenn ich mich in Begrifflichkeit
einlasse, so schwimme ich und
fühle mich als Schwätzer,
wenn
wobei es mich nicht erleichtert, der andere
auch ein Schwätzer ist, und das kommt vor,

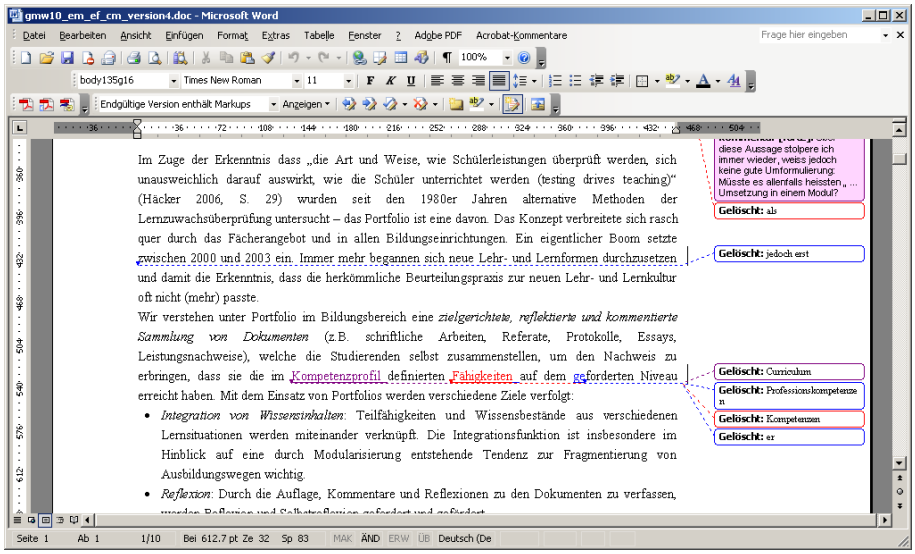
Abbildung 2.10: Ausschnitt einer Manuskriptseite aus dem Tagebuch von Max Frisch [Frisch 2010, S. 177].

Zusätzlich finden wir Änderungen, die in einem expliziten Redigierprozess angebracht wurden, etwa die komplette Streichung eines grossen Teils des Textes auf diesem Blatt – eine solche Markierung ist nicht möglich, wenn das Blatt noch in der Maschine eingespannt ist, die Streichung der ersten Zeile

mit einem Stift wäre hingegen auch bei noch eingespanntem Blatt möglich. Diese Anmerkungen und Redigieranweisungen müssen dann in einem nächsten Durchgang des Schreibens des Textes integriert werden und führen so zu einer zweiten Fassung.

Abbildung 2.11 zeigt das Redigieren eines Textes, der mit *MS Word* erstellt wird – die Änderungen werden gekennzeichnet, Änderungen verschiedener Autoren werden farblich unterschieden. Es ist möglich, die Kennzeichnung der Änderungen auszublenden, sodass nicht mehr sichtbar ist, wer was an der gerade sichtbaren Fassung eines Textes geändert hat.

Abbildung 2.11: Redigieren eines Textes mit MS Word *unter Verwendung der Option* `Änderungen nachverfolgen`.



Flinn [1987], Piolat [1991] und auch Hill et al. [1991], stellen fest, dass *erfahrene Autoren* innerhalb aller Ebenen eines Textes – also sowohl auf der Oberfläche linguistischer Einheiten als auch auf struktureller Ebene – redigieren und bereit sind, bereits Geschriebenes zu verwerfen und neu zu schreiben oder komplett umzustellen. Erfahrene Autoren beziehen also *Makrostrukturen* ein, siehe [Flinn 1987, S. 36ff], [Piolat 1991, S. 261] und [Hill et al. 1991, S. 101]. Eher *unerfahrene oder ungeübte Autoren*[15] redigieren lediglich an der Oberfläche, also *Mikrostrukturen*. Sie beschränken sich meist auf die Korrektur von Orthographie- und Grammatikfehlern, siehe [Flinn 1987, S. 39ff], [Piolat 1991, S. 255] und [Piolat et al. 1997, S. 567]. Sie lassen sich von Regeln leiten, die sie strikt umsetzen [Hill et al. 1991, S. 100].

Textbearbeitungsprogramme unterstützen grundsätzlich beide Redigierstile bzw. Benutzergruppen [Flinn 1987, S. 41]. Piolat [1991, S. 258] stellt fest, dass die Arbeit mit Textbearbeitungsprogrammen zwar zu mehr Revisionen führten, diese jedoch nicht zwingend in die Tiefe gingen. Gleiches bestätigt Severinson Eklundh [1994, S. 204ff] einige Jahre später. Wir finden für das Redigieren eines Textes also ebenso elementare Unterschiede zwischen Anfängern und erfahrenen Autoren, wie Andriessen et al. [1996, S. 253] sie für das Planen des Textes angeben.

Das Redigieren elektronisch erstellter Texte schliesst Medienbrüche ein: Texte werden mit Textbearbeitungsprogrammen geschrieben, dann ausgedruckt, auf Papier werden Redigieranweisungen festgehalten, die dann wieder mit dem

15. «Ungeübt» bezieht sich auf das Schreiben qualitativ hochstehender Texte, nicht auf den Umgang mit einem bestimmten Schreibwerkzeug.

Textbearbeitungsprogramm umgesetzt werden. Wird nur elektronisch gearbeitet, stellt sich die Frage: «Is text revision, and thus text improvement, clearly facilitated by the use of word processors?» [Piolat 1991, S. 255]

Zu Beginn der 1980er Jahre wird in Studien, wie von Daiute [1983], das Redigieren am Computer ausschliesslich als Vorteil zur Überwindung von psychologisch begründeten Vorbehalten und tatsächlichen Schwierigkeiten beim Redigieren überhaupt gesehen:

> When writers make corrections by hand, they have trouble considering them as an integral part of the piece. With the text editor, writers can have clean copies of evolving texts at any time. Thus, they can carefully consider the new piece as a whole and evaluate the overall effects of each change.
>
> This physical ease of revising encourages writers to experiment and to view their writing as dynamic. [Daiute 1983, S. 137][16]

Die Dissertation von Lutz [1983] widmet sich explizit der Frage, wie sich das Redigieren am Computer vom Redigieren auf Papier unterscheidet. Sie führt verschiedene Experimente und Interviews mit erfahrenen Autoren durch, die teilweise beruflich schreiben, etwa Journalisten. Am Computer sind mehr Bewegungen im Text festzustellen, die bearbeiteten Textstellen liegen jedoch nah beieinander. Die meisten Änderungen betreffen Oberflächen linguistischer Einheiten; die Zahl der Änderungen je linguistischer Einheit – Wort, Wortgruppe, Satzglied, Satz – nimmt mit der Grösse der Einheit ab. Satzglieder werden also seltener verändert als einzelne Wörter. Da in der Regel Textstellen redigiert werden, die im Blick des Autors sind, korreliert die Weite der Änderungen mit der Bildschirmgrösse – also dem Schreibmedium –, siehe [Lutz 1983, S. 204].

Wenn auch die Anzahl der durchgeführten Redigieroperationen oder Änderungen keinen Hinweis auf die Qualität des schliesslich «fertigen» Textes gebe, plädiert Lutz für den Einsatz des Computers, da er mehr und schnellere Redigieroperationen erlaube und so den kreativen Aspekt des Redigierens unterstütze: «[…] revisions themselves were often revised in an everevolving pattern of choices to achieve the most acceptable presentation of ideas.» [Lutz 1983, S. 241] Lutz empfiehlt jedoch das Redigieren auf Papier – also den Medienbruch –, da Blätter erlaubten, gleichzeitig betrachtet zu werden, vorzunehmende makrostrukturelle Änderungen seien so leichter zu markieren [Lutz 1983, S. 243]. Tatsächlich stellt Hawisher [1988] fest, dass Texte zwar mit Hilfe von Textbearbeitungsprogrammen erstellt, dann jedoch ausgedruckt und Redigiermarkierungen und -anweisungen auf Papier festgehalten würden, da dies übersichtlicher sei [Hawisher 1988, S. 9f]. Anschliessend wird am Computer der bisherige Text entsprechend den Anmerkungen überarbeitet und eine neue Fassung erstellt.

Hawisher [1986] fasst die bis dahin 24 publizierten Studien zum Schreiben am Computer zusammen und konstatiert:

> Although in all these studies final drafts exhibited fewer errors, committing and correcting more surface errors is different from

16. Daiute verwendet hier die Bezeichnung *text editor*, meint jedoch *word processor*, wie bereits in Daiute und Taylor [1981].

> making fewer errors. Increased attention to errors, for example, might well detract from students' thinking, thereby lowering the overall quality of a piece of writing. [Hawisher 1986, S. 16]

Das Redigieren am Computer entsprach also nicht den in Abschnitt 2.3.1 vorgestellten Definitionen von Redigieren, sondern lediglich einer Fehlerkorrektur. Das Fokussieren auf das Finden und Korrigieren von Fehlern behinderte inhaltliche Änderungen, die notwendig gewesen wären.

Zwei Jahre später sind weitere sechzehn Studien erschienen, die jedoch gesamthaft nicht zu einer Klärung der Frage nach dem Einfluss des Computers auf das Redigieren beitragen. Es lässt sich feststellen, dass mehr Revisionen beim Redigieren mit dem Computer im Vergleich zum Redigieren auf Papier durchgeführt werden. Die Studien liefern sowohl Belege dafür, dass daraus eine bessere Qualität der endgültigen Texte resultiere, als auch dafür, dass hinsichtlich der Qualität keine Unterschiede festzustellen sei, siehe [Hawisher 1988, S. 7]. Interessant ist diese Anmerkung:

> Interestingly, none of these studies examined the different kinds of errors, that might be characteristic of writing with computers—having the same phrase, for example at both the start and end of a sentence because the writer forgot to delete one. [Hawisher 1986, S. 16]

Ein solcher Fehler entsteht vermutlich durch das Umstellen eines Satzes durch Kopieren oder Verschieben von Satzteilen und kann als *slip* nach Norman [1981] verstanden werden. Da solche Operationen auf Papier nicht möglich sind, werden dort auch solche Fehler nicht auftreten. Derartige Fehler können natürlich einerseits *erkannt* werden durch den Einsatz von automatischen Prüfwerkzeugen, die den entstandenen Text auf Wohlgeformtheit überprüfen, und andererseits *vermieden* werden durch optimale Funktionen, die den Autor bei seiner Redigierarbeit unterstützen.

Piolat [1991] stellt fest, dass der Unterschied zwischen Anfängern und erfahrenen Autoren – bezogen auf die Arbeitsweise und auch auf die Qualität des endgültigen Textes – grösser würde, wenn das Schreibwerkzeug nicht Papier und Stift, sondern der Computer sei [Piolat 1991, S. 265]. Erfahrene Autoren seien in der Lage, gleichzeitig – also in einem Durchgang durch den Text – inhaltliche Änderungen vorzunehmen *und* linguistische Mängel eines Textes zu beheben, während Anfänger dies in getrennten Durchgängen angingen [Piolat 1991, S. 266]. Einige Jahre und einige Studien später fassen Piolat et al. ihre Erkenntnisse so zusammen:

> Writing with a word processor has been found to hinder revising in several ways. Error detection on the screen is slower and fewer errors are detected. […] Participants also take more time finding a piece of information on the screen (even if the text is bigger than one page) than on a printed copy […] They rarely use the word searching function, even though it would speed up their work and confine revisions to the highlighted portion of the text […] Although the absolute number of revisions is larger when revisions are made on a word processor, their corrections are usually

> surface corrections (letters, words, punctuation) and, unlike paper and pencil writing they hardly ever affect deeper levels like text coherence and overall organization […]. [Piolat et al. 1997, S. 567]

Die Interviews von Dowling [1994] mit erfahrenen Autoren bestätigen dies. Textbearbeitungsprogramme mit ihrem Anspruch des *What-you-see-is-what-you-get* (WYSIWYG) böten zuviele Möglichkeiten, *während* des Schreibens das Layout zu verändern, und behinderten so das Schreiben: Der Text sehe einerseits selbst in der Rohfassung vollkommen aus und diese «Perfektheit» verhindere editorische Eingriffe. Auf der anderen Seite sei ein Text in einem Textbearbeitungsprogramm immer in Bearbeitung. Es sei schwierig, einen Endpunkt des Schreibens und Redigierens festzulegen [Dowling 1994, S. 230ff]. Wir finden hier die negativen Auswirkungen der Vermischung der zwei Sichtweisen von *Text*. Während es notwendig wäre, tatsächlich den bislang geschriebenen Text im linguistischen Sinne (*text produced so far*) zu evaluieren, wird dem Autor das bislang produzierte *Dokument* präsentiert. Rein inhaltliche Überlegungen vermischen sich mit ästhetischen Aspekten, die Änderung der Wortwahl ist ebenso einfach möglich wie die Änderung der Schriftgrösse. Zudem seien in einem WYSIWYG-Editor Strukturen und Auszeichnungen nicht sichtbar, somit nicht direkt veränderbar, jedoch unbeabsichtigt von anderen Änderungen betroffen, wie Holmes [2001] feststellt.

Severinson Eklundh [1992] bemerkt in der Zusammenfassung zu Studien zum Redigieren:

> […] there is a conflict inherent in the medium: while the word processor encourages early on-screen drafting and continuous revision of the written text, it simultaneously hampers the ongoing evaluation of the changes with regard to global aspects of the written text. [Severinson Eklundh 1992, S. 74]

Diese Einschätzung hat sich also über die Jahre und mit der Weiterentwicklung von Textbearbeitungsprogrammen nicht geändert: Obschon die Textbearbeitungsprogramme aus den 1970er und 1980er Jahren heute gar nicht mehr existieren, sind ihre Eigenschaften hinsichtlich des Schreibprozesses denen der jeweils aktuellen Programme verblüffend ähnlich. Lutz konstatiert schon 1983:

> This finished quality may also mean, however, that work on a text is ended prematurely, before it has been revised to satisfy the needs of writer and reader. […] If a pen and paper writer wishes to reorganize a text, he must literally cut and paste or recopy it. Computer writers may insert, delete, and substitute lines or paragraphs, and the spacing of the text automatically adjusts to accommodate these changes without spoiling the appearance of the finished text. [Lutz 1983, 137]

Dieser «schöne» optische Eindruck eines Textes während des gesamten Schreibprozesses wird zu Beginn der 1980er Jahre hauptsächlich als positiv angesehen und als willkommener Effekt, um vor allem das Redigieren zu erleichtern:

> The text editor also eliminates the spatial and aesthetic barriers that are special inhibitators of revising activity. Writers are often reluctant to mess up a carefully written page by crossing out words or cramping inserts metween lines and in the margins. [Daiute 1983, S. 136][17]

Holdstein und Redman gehen von Berichten aus, die bei der Verwendung des Computers mehr Schreib- und Redigieraktivitäten als bei der Verwendung von Stift und Papier beschreiben, finden dies in ihren eigenen Experimenten jedoch nicht bestätigt. [Holdstein und Redman 1985, S. 52]. 20 Jahre später betont Merz-Grötsch [2005] die «neuen» Möglichkeiten des Einsatzes des Computers als Schreibwerkzeug. Es ergäben sich neue Arbeitsformen, da es sehr leicht sei, Textpassagen zu verschieben, zu kopieren, Texte oder Fragmente zu speichern und zu formatieren und zusätzliche Hilfsmittel zu benutzen, etwa Wörterbücher oder Korrekturprogramme.

> Ein grosser Vorteil der Textproduktion am Computer liegt darin, dass fehlerhafte Wörter oder holprige Formulierungen, Auslassungsfehler oder gar ganze Textpassagen, die an der falschen Stelle stehen, im Handumdrehen verbessert und überarbeitet werden können. [Merz-Grötsch 2005, S. 71]

Ähnlich argumentieren Eyman und Reilly:

> One of the key features of word processing applications is the ability to select text of any length (word, sentence, paragraph, page), remove it from its current location in a document, and place it in another location. [Eyman und Reilly 2006, S. 105]

In all diesen Studien wird jedoch nicht reflektiert, ob und wie die Probanden eine Unterweisung zur Benutzung der Textbearbeitungsprogramme erhalten.

Wird ausschliesslich mit dem Computer gearbeitet und kein Papierausdruck zum Redigieren verwendet, lässt sich feststellen, dass – im Vergleich zum Schreiben ausschliesslich auf Papier – vermehrt «während» des Schreibens redigiert wird und nicht in einer expliziten Redigierphase [Severinson Eklundh 1994, S. 204f]. Das Schreibwerkzeug ändert also den Prozess, nicht unbedingt das Produkt.

Die Suche nach Fehlern an der Oberfläche eines Textes, also auf Ebene von Grammatik und Orthographie, wird von Werkzeugen zur automatischen Überprüfung in Textbearbeitungsprogrammen unterstützt. Es existieren diverse Prüfwerkzeuge, sowohl als eigenständige Programme als auch integriert in Editoren.

Bei der Behebung von gefundenen Fehlern wird der Autor relativ gut unterstützt, sofern eine Fundstelle tatsächlich einen Fehler beinhaltet und die gestellte Diagnose richtig ist. Wir gehen in Abschnitt 3.3 noch genauer darauf ein. Bei der Überarbeitung eines Textes, die über die Korrektur von orthographischen Fehlern hinausgeht und den Text auch inhaltlich verändert, erhält ein Autor dagegen wenig Unterstützung. Zu denken ist dabei an die Veränderung der

17. Daiute [1983] spricht hier von *text editor*, bezieht sich jedoch tatsächlich nicht auf Programmiereditoren, sondern auf Textbearbeitungsprogramme.

Wortstellung, das Austauschen oder Verschieben von Wortgruppen etc. Also Aktionen, die sich auf «intellektuelle» – im Gegensatz zu den «automatischen» Entscheidungen eines Prüfwerkzeugs – Entscheidungen zur Veränderung des Textes stützen und dem kreativen Aspekt des Redigierens Rechnung tragen.

Studien, die sich mit dem Redigieren beschäftigen, beobachten Autoren unterschiedlichen Alters, mit unterschiedlichen Fähigkeiten und berichten über die beobachteten Phänomene. Sie geben Empfehlungen zur Verbesserung des Prozesses und zur Entwicklung von Schreibstrategien, ohne auf die Möglichkeiten des Schreibwerkzeugs näher einzugehen. Nur wenige, wie etwa Eyman und Reilly [2006], beziehen die technischen Möglichkeiten von Textbearbeitungsprogrammen ein und schlagen die explizite Nutzung der Funtionalitäten vor:

> An instructor can ask students to change active verbs to boldface, highlight passive constructions in italics, use large fonts for descriptive words, underline the thesis statement, or select particular font color for topic sentences in each paragraph. This kind of visual marking presents a striking image of the text and can show the writer elements that may be overused or missing. [Eyman und Reilly 2006, S. 106]

Wie für den Schreibprozess allgemein wurden auch für das Redigieren verschiedene Studien durchgeführt, um die kognitive Belastung zu ermitteln. Collier [1983], Piolat [1991] und Piolat et al. [2004] zeigen (jeweils im Abstand von etwa 10 Jahren), dass die kognitiven Ressourcen durch das Redigieren am Computer stark beansprucht werden:

> […] several sequential keyboard moves are frequently required to accomplish but one alteration in a text, and I saw after viewing only a few minutes of the videotapes that the attention shifts needed when a writer is manipulating the keyboard for operations other than simple typing interrupted continuous concentration on the text itself. [Collier 1983, S. 153]

Je nach Erfahrung oder Übung der Autoren werden die kognitiven Ressourcen unterschiedlich auf verschiedene Subprozesse des Redigierens verteilt. Insgesamt bleibt die Belastung jedoch für alle sehr hoch [siehe Piolat et al. 2004, Torrance und Galbraith 2006]. Diese Befunde korrespondieren mit den Erkenntnissen zur kognitiven Belastung des Schreibprozesses insgesamt, wie in Abschnitt 2.2 schon gezeigt.

2.3.3 Klassifizierung von Redigieroperationen

Autoren redigieren Texte auf mehreren Ebenen und hinsichtlich verschiedener Kriterien:

- Änderungen werden im *Gesamttext*, innerhalb von *Absätzen*, auf *Satz*ebene und auf *Wort*ebene vorgenommen [Mißler 1997, S. 165] [Schmitz und Zöllner 2006, S. 87].

- Es können *globale* – Struktur und Inhalte betreffende – und *lokale* – Wortwahl, Grammatik und Orthographie betreffende – Änderungen unterschieden werden [Mißler 1997, S. 165].
- Änderungen betreffen die *äussere Form* oder den *Stil* eines Textes [Mißler 1997, S. 169].
- Blatt unterscheidet das *Korrigieren von Fehlern*, das *Emendieren von Ausdruck und Stil*, das *Verändern von Textpassagen* (als Redigieren im engeren Sinne) und das *Neufassen von Textpassagen* [Blatt 2004, S. 45].
- Dale unterscheidet *inhaltliche* Änderungen und *strukturelle* Änderungen. Er hält fest, dass die meisten dieser Arbeiten nur von Menschen erledigt werden könnten, automatische Unterstützung sei nur in sehr geringem Masse möglich [Dale 1990, S. 69].
- Lutz unterscheidet *revising* als die Änderungen, die ein Autor an eigenen Texten vornimmt, und *editing* als die Änderungen, die an Texten anderer Autoren vorgenommen werden [Lutz 1983, S. 115].
- Allal und Chanquoy unterscheiden zwischen *editing* – der Fehlerkorrektur – und *rewriting* – inhaltlichen Änderungen [Allal und Chanquoy 2004, S. 2].

Die Änderungen, die Obama in seiner Rede vorgenommen hat (siehe Abschnitt 2.3.1), können wir entsprechend beschreiben. Neben den oben genannten groben Unterscheidungen von Redigieroperationen bzw. beobachtbaren Textveränderungen finden sich in der Literatur auch explizite Taxonomien mit Klassifizierungen auf mehreren Ebenen. Diese Taxonomien von Redigieroperationen beruhen zumeist auf Analysen verschiedener Versionen von Texten, wie sie von Autoren in Studien explizit verlangt oder während des Schreibens automatisch archiviert wurden. In den verschiedenen Versionen von Texten werden Änderungen dann unter Verwendung solcher Taxonomien annotiert und entsprechend ausgewertet. Wir stellen im Folgenden die verbreitetsten Taxonomien vor:

- Das Klassifikationsschema von Bridwell [1980] stützt sich auf 100 Vorversionen von Texten von Schülern, die auf Papier einen Text schreiben und diesen – jeweils mit einem anderen Stift – zweimal überarbeiten sollten. Die so extrahierten Klassen von Änderungen konzentrieren sich auf die Oberfläche des Textes und nehmen keinen Bezug zu mentalen Prozessen der Autoren. Hier wurden nur die gesammelten Texte ausgewertet. Bridwell unterscheidet die Änderungen hinsichtlich der involvierten linguistischen Strukturen: (a) Oberfläche von Wortformen (*surface level*), (b) lexikalische Ebene (*lexical level*), (c) Wortgruppen (*phrase level*), (d) Teilsätze (*clause level*), (e) Sätze (*sentence level*), (f) Folgen von Sätzen (*multi-sentence level*) und (g) Gesamttext (*text level*) [Bridwell 1980, S. 203f]. Innerhalb dieser Klassen wird dann nach den vorgenommenen Änderungen (etwa Hinzufügen, Löschen, Umordnen, Ändern der Gross-/Kleinschreibung) unterschieden.
- Sommers [1980] dagegen klassifiziert auf zwei Ebenen und unterscheidet die Ebene der Redigieroperationen – Löschen (*deletion*), Ersetzen (*substitution*), Hinzufügen (*addition*), Umordnen (*reordering*) – und die

linguistische Ebene, auf der eine Änderung vorgenommen wird – Wort (*word*), Wortgruppe (*phrase*), Satz (*sentence*), Thema (*theme*) [Sommers 1980, S. 380]. Die Taxonomie von Sommers stützt sich nicht auf die Arbeiten von Schülern, sondern auf Schreibprozesse von Erwachsenen und erfahrenen Autoren.

- Nur wenig später entwickeln Faigley und Witte [1981] eine Taxonomie und kritisieren die bereits vorhandenen als unzureichend, da deren Kategorien sich überlappen würden. Auch Faigley und Witte wollen ihre Taxonomie als Grundlage zur Annotierung von Texten in verschiedenen Stadien verwenden. Sie beziehen nicht linguistische Strukturen ein, sondern orientieren sich an der Bedeutung des Textes: « Our taxonomy of revision changes is based on *whether new information is brought to the text or whether old information is removed in such a way that it cannot be recovered through drawing inferences*.» [Faigley und Witte 1981, S. 402][18] Unterschieden werden Änderungen, die keine Veränderung der Bedeutung des Textes verursachen (*surface changes*), und Änderungen, die eine solche Veränderung bewirken (*textbase changes*). Erstere können linguistische, genauer morphosyntaktische, Eigenschaften von Elementen umfassen sowie in geringem Umfang auch Hinzufügungen, Löschungen, Ersetzungen und Umordnungen. Diese können als Paraphrasierungen angesehen werden. Für bedeutungsändernde Operationen wird die betroffene Ebene unterschieden. Hinzufügungen, Löschungen, Ersetzungen und Umordnungen können sowohl auf Makroebene (den ganzen Text betreffend) als auch auf Mikroebene (mit nur lokaler Auswirkung) erfolgen. [Faigley und Witte 1981, S. 402ff]

- In der Definition des Redigierprozesses von Fitzgerald [1987] ist ebenfalls eine Klassifizierung von Redigieroperationen enthalten:

  > Changes may or may not affect meaning of the text, and they may be major or minor. Also changes may be made in the writer's mind before being instantiated in written text, at the time, text is first written, and/or after text is first written. [Fitzgerald 1987, S. 484]

- Allal und Chanquoy [2004] nennen drei Kriterien, nach denen Revisionen kategorisiert werden können. Wir finden hier wie schon bei Faigley und Witte die Berücksichtigung von Elementen, die mit veränderten Auffassungen und Definitionen des Redigierprozesses einhergehen. Insbesondere werden Operationen berücksichtigt, die sich nicht auf schon geschriebenen, sondern erst noch zu schreibenden Text beziehen, die also an der Oberfläche des Textes nicht beobachtet werden können:

  > (1) pretextual revision, which affects intentions, plans or mental formulations of text before transcriptions has occured; (2) on-line revision, which is integrated in the process of transcription and entails changes made while reviewing a word or group of words that has just been written; (3) deferred revision which takes place once a relatively complete draft—of a text or a sizeable part of a text (e.g. a chapter)—has been written. [Allal und Chanquoy 2004, S. 2]

18. Hervorhebung im Original.

with keystroke logging data. Results showed that pre-contextual revisions affected the text written thus far both on a formal and on a conceptual level.

The category 'contextual revision' is divided into two sub-categories: Form and Conceptual. The first category, Form, include revisions that do not affect the content of the text. Form revisions are further defined as 'conventional', if the grammar of the language requires the revision, or 'optional' if it does not and 'correct', 'erroneous' or 'neutral' according to whether the revision corrects or creates a mistake or does neither (Allal, 2000; Chanquoy, 2001). Conceptual revisions affect the content of the text, and include 'text-based' revisions, that are triggered by the text, and 'balance' revisions, that are triggered by the writer's style or audience orientation.

- Lindgren [2005] klassifiziert in ihrer Dissertation ebenfalls Redigieroperationen. Unterschieden wird zwischen *contextual* und *pre-contextual* (Revisionen, die noch im Kontext stattfinden, aber vermutlich ausgeschlossen wird). Tatsächlich beobachtbare Revisionen werden dann noch einmal bezüglich Inhalt und Form unterschieden, siehe Abbildung 2.12.

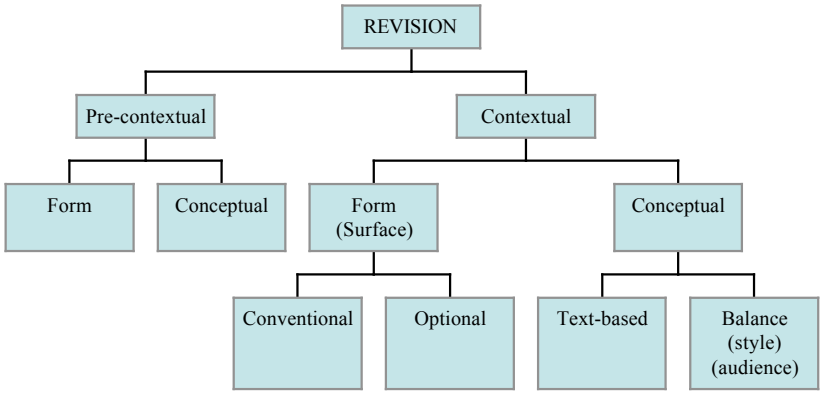


Figure 2. Overview of the revision taxonomy.

Abbildung 2.12: Taxonomie von Redigieroperationen von Lindgren [2005, S. 20].

Lindgren bezieht sich ausdrücklich auf die Taxonomie von Faigley und Witte, die ihre Taxonomie unter anderem damit motiviert hatten, dass alle bislang verfügbaren Taxonomien überlappende Kategorien aufwiesen. Leider behält Lindgren diesen Aspekt nicht bei, sondern führt Unterkategorien ein (*form* und *conceptual*), die in jeder der Hauptkategorien vorkommen. So ist es dann auch möglich, dass Lindgren und Sullivan [2006] die gleichen Kategorien nach einem anderen Prinzip darstellen und *form* und *conceptual* als Hauptkriterien verwenden, siehe [Lindgren und Sullivan 2006, S. 172ff].

In der Gesamtschau fällt auf, dass immer wieder Vorschläge zu neuen oder abgewandelten Taxonomien präsentiert werden, die auf bereits bekannten Klassifizierungen beruhen und sich auf Erkenntnisse aus neuen Studien stützen. Die vorgeschlagenen Kategorien berücksichtigen zudem veränderte Sichtweisen auf den Schreib- und damit auch auf den Redigierprozess. Jedoch basieren alle Taxonomien auf Beobachtungen, die sich in verschiedenen Versionen eines Textes, also im *Produkt*, manifestieren. Die vorhandenen Taxonomien sind also – im Gegensatz zu den in Abschnitt 2.1.1 präsentierten Modellen des Schreibprozesses als Ganzem – produkt- statt prozessorientiert. Dies ist damit begründbar, dass der Prozess selbst nicht direkt beobachtbar ist – verzichtet man auf die Schreibsituation verändernde Eingriffe, wie lautes Denken. Wie für Schreibmodelle spielt in Redigiertaxonomien das verwendete Werkzeug keine Rolle. In den letzten Jahren werden jedoch für elektronisch produzierte Texte die Möglichkeiten der Arbeitsumgebungen genutzt und so gewonnene Erkenntnisse in Taxonomien und Definitionen integriert. Wir gehen darauf im nächsten Abschnitt ein.

2.3.4 Aufzeichnung von Redigieroperationen

Forschung zum Redigieren kann sich nicht nur auf den Abgleich von Textfassungen zu verschiedenen Zeitpunkten stützen, wie es etwa Lutz [1983] durchführt. Es ist notwendig, den Prozess zu beobachten, wie Hill et al. [1991] argumentieren. Wird ein Text am Computer erstellt und bearbeitet, kann aufgezeichnet

werden, welche Tasten gedrückt wurden – sogenanntes *keystroke-logging*. Redigieroperationen werden dann aus Mustern von Folgen einzelner Tastendrücke erschlossen.

S-Notation [Kollberg 1998] ist eine Möglichkeit, Redigieroperationen zu beschreiben und dabei von der blossen Abfolge von Tastendrücken oder Maus-Bewegungen zu abstrahieren. Folgen von Tastendrücken werden automatisch klassifiziert. Die Abfolge von Redigieroperationen kann unabhängig vom verwendeten Schreibprogramm automatisch aus dem Keystroke-Logfile erzeugt und wieder abgespielt werden. Alle nicht schreibbezogenen Tastendrücke werden entfernt, siehe [Severinson Eklundh 1994, S. 208] und [Kollberg und Severinson Eklundh 2002, S. 91].

Die Klassifizierung der beobachteten Operationen beschränkt sich auf die Differenzierung zwischen *Einfügen* und *Löschen* [Kollberg und Severinson Eklundh 2002, S. 92]; das *Ersetzen* von Elementen wird durch eine Kombination aus Löschen und Einfügen kodiert [Severinson Eklundh 1994, S. 208]. In der S-Notation wird neben diesen beiden Auszeichnungen weiterhin festgehalten, in welcher Reihenfolge und an welcher Position etwas eingefügt oder gelöscht sowie wann und wo mit dem Schreiben fortgefahren wurde. [Kollberg und Severinson Eklundh 2002, S. 92] Eine Abstraktion auf höherer Ebene erfolgt explizit nicht:

> An important requirement on a formal revision notation is that it should be neutral with respect to the writers' intentions, and thus not make any assumptions as to the purpose of a certain change in the text. […] For example, if a writer deletes a word, and subsequently inserts another word at the same position in the text, one cannot deduce that the writer intended the second word to replace the first (even if this is often the case). [Kollberg und Severinson Eklundh 2002, S. 92]

Um zu ermitteln, welche Redigieroperation ausgeführt wurde, können die tatsächlichen Tastendrücke und die *Pausen* zwischen Tastendrücken verwendet werden. Aus solchen Daten ist jedoch nicht eindeutig zu ermitteln, ob Pausen *vor* einer Operation oder dem Schreiben eines Wortes oder *nach* einer solchen Operation erfolgen, also eher dem Planen oder dem Redigieren zuzuordnen sind [siehe Severinson Eklundh und Kollberg 1996, S. 181]. Ebenso kann nicht ermittelt werden, ob das Löschen eines gerade geschriebenen Buchstabens das Korrigieren eines Tippfehlers oder die Revision der ursprünglich geplanten Fortsetzung des Satzes ist.

Die Ableitung von abstrakteren und intendierten Redigieroperationen kann dann erst durch die Interpretation der mittels S-Notation annotierten Daten erfolgen, siehe [Kollberg und Severinson Eklundh 2002, S. 93] und [Severinson Eklundh und Kollberg 1996, S. 182]. Eine Variante davon nennen Kollberg und Severinson Eklundh *Redigierepisoden* (engl. *episodes of revision*). Dies ist jedoch eine Abstraktion auf sehr hoher Ebene. Die Kriterien für die drei Klassen von Redigierepisoden sind der Ort der Revision und die Verschränkung von Revisionen. Unterschieden werden (a) Episoden mit mehreren Revisionen an der gleichen Cursor-Position, (b) Revisionen, die durch eine weitere Revision unterbrochen werden, und (c) Unterbrechung des aktuellen Schreibvorgangs, um eine bereits geschriebene Passage zu redigieren und anschliessend an der

Unterbruchstelle weiterzuschreiben. [Kollberg und Severinson Eklundh 2002, S. 94f]

Diese Unterscheidung ist natürlich sehr grob und berücksichtigt die Intention des Autors nicht. Die Absichten eines Autors, die einzelnen Veränderungen eines Textes zugrundeliegen, können durch blosses Beobachten oder Aufzeichnen der Tastendrücke nicht identifiziert werden:

> […] the most salient problem with keystroke records is that the revisions made by the writer are not directly accessible in the record. Both the exact nature of a change and the place in the text that is affected often have to be inferred by detailed manual analysis. [Severinson Eklundh 1994, S. 208]

Obwohl oft betont wird, dass Schreiben nicht das Aneinanderreihen von Buchstaben, sondern die Bearbeitung linguistischer Einheiten ist (siehe etwa [Daiute und Taylor 1981] oder [Bolter 1989]), wird bei der Auswertung der *keystroke-logging*-Daten Veränderung auf Zeichenebene annotiert und untersucht. Streng genommen wird hier nicht der Prozess, sondern das Produkt beobachtet.

Nach Einschätzung von Kollberg und Severinson Eklundh können S-Notation-Daten die Grundlage zur Analyse des Redigierens eines Textes bilden, müssen jedoch mit weiteren Informationen ergänzt werden, um die intendierten Redigieroperationen auf einer höheren Ebene zu ermitteln: «Arriving at such a description often requires access to information outside of the computer records, such as verbal reports from the writer.» [Kollberg und Severinson Eklundh 2002, S. 99] Hier finden wir jedoch das Problem, das wir bereits in Abschnitt 2.1.2 geschildert haben – lautes Denken beeinträchtigt den Schreibprozess und damit auch das Redigieren. Die Entwicklung von Programmen, «[…] in which S-Notation records of a session are built upon to yield preliminary, local interpretations of certain revising patterns.» [Kollberg und Severinson Eklundh 2002, S. 100] erfolgte nie[19], auch wenn dies sehr attraktiv wäre:

> Such an analysis might draw tentative conclusions from sequences of elementary revisions about their purpose in a context, based on statistics, keystroke data or previous research. This would be an alternative to the revsion analyses made by many researchers, where interpretations are based on intuition alone. [Kollberg und Severinson Eklundh 2002, S. 100]

Weder [2010] verwendet *retrospektive Interviews* (engl. *stimulated recall*) [DiPardo 1994], eine Technik ähnlich der gelenkten Retrospektion: Sie befragt die Probanden gezielt zu einzelnen Situationen während der Schreibsitzung, die sie ihnen als Videoaufnahme vorspielt. Die Aussagen der Probanden werden verwendet, um die aufgezeichneten Veränderungsprozesse im Text interpretieren zu können.

Perrin entwickelt die *Progressionsanalyse* (PA), um zusätzlich zu aufgezeichneten Tastendrücken für den Schreibprozess relevante Information zu protokollieren und in der Analyse zu berücksichtigen, siehe [Perrin 2002, 2006a,b]. Mit

19. Persönliche Kommunikation mit Kerstin Severinson Eklundh, E-Mail vom 9. Juni 2008, Message-ID `<2BD74CC7-CB76-4412-BE34-AF89413C2245@csc.kth.se>`.

Hilfe der Progressionsanalyse können Strategien entwickelt werden, die professionellen Autoren das Schreiben erleichtern. Die Ausgabe der *S-Notation* wird erweitert, sodass Ergänzungen, die nicht geschrieben, sondern mittels `copy-paste` aus anderen Texten oder von anderen Stellen im selben Text eingefügt wurden, gesondert ausgezeichnet werden [Perrin 2002, S. 109]. Während *S-Notation* die Entwicklung eines Textes in diesem Text selbst darstellt, ermöglichen Progessionsdiagramme eine graphische Darstellung der Redigierepisoden als Weg durch den Text. So kann leicht festgestellt werden, ob einzelne Passagen mehrfach redigiert wurden. [Perrin 2002, S. 110ff] Da dies allein, wie auch Kollberg und Severinson Eklundh [2002] feststellen, nicht ausreicht, um den Redigierprozess möglichst vollständig zu erfassen, wird anschliessend – mit so geringem Abstand zum Zeitpunkt des Schreibens wie möglich – die Aufzeichnung der Tastendrücke dem Autor vorgespielt und dieser kommentiert die einzelnen Vorgänge. [Perrin 2002, S. 113ff] Somit werden Beeinträchtigungen des Schreibprozesses durch lautes Denken vermieden und es ist möglich, Schreibprozesse im realen Arbeitsumfeld von Autoren zu untersuchen – das Aufzeichnen der Tastendrücke erfolgt vom Autor unbemerkt im Hintergrund und hat keinen Einfluss auf den Schreibprozess oder auf die Geschwindigkeit o. ä. des Computers.

Ähnlich geht schon in den 1980er Jahren Flinn [1987] vor. Sie nimmt ein Jahr lang die Tastendrücke während des Schreibens mit *COMPTRACE* auf, einer modifizierten Version des *Milliken Word Processors* [Smithson 1986], und bespricht die Aufnahmen anschliessend mit dem jeweiligen Autor. Diese Interviews werden ebenfalls aufgezeichnet. Ein Grund für dieses Vorgehen ist explizit die Vermeidung von Unterbrüchen im Schreibprozess durch lautes Denken. [Flinn 1987, S. 33ff]

In Studien, die Programme wie *JEdit*, *ScriptLog* oder *Translog* (siehe Van Waes et al. [2006] und Sullivan und Lindgren [2006] für einen Überblick) für *keystroke-logging*-Studien verwenden, werden die tatsächlichen Tastendrücke oder Funktionsaufrufe nicht ausgewertet, sondern nur die *Veränderungen*, die sich durch Tastendrücke, Maus-Bewegungen und Funktionsaufrufe an der *Textoberfläche* manifestieren. Die Ebenen der Textoberfläche und der Verwendung des Schreibwerkzeuges werden nicht in Beziehung zueinander gesetzt. Es wird nicht ermittelt und annotiert, *wie* eine Wortform gelöscht und durch eine andere ersetzt wurde, sondern nur, *dass* es passiert ist.

Erst in den letzten Jahren interessieren sich Schreibforscher wieder für Fragen, die über die Veränderung des Textes hinausgehen. Mit *Inputlog*[20] [Leijten und Van Waes 2005a] ist es möglich, *alle* Tastendrücke und Maus-Bewegungen aufzuzeichnen sowie die exakten Zeitpunkte und die Veränderung der Textoberfläche damit zu verbinden. Es wurde eine XML-basierte Auszeichnung entwickelt, um Redigieroperationen auf der Textoberfläche auszeichnen zu können. [Van Waes et al. 2009, S. 45]

In einem gerade begonnen Projekt, «Merging writing process data with lexica», sollen die Textveränderungen (möglichst automatisch) auch auf linguistischer Ebene ausgezeichnet werden.[21] Da die entsprechenden Daten zur

20. Siehe auch Projektwebsite `http://www.inputlog.net/` (zuletzt besucht am 8.12.2010, 19:34).

21. «Merging writing process data with lexica», Project Details, online `http://www.ua.ac.be/main.aspx?c=luuk.vanwaes&n=14&pid=23965&more=0` (zuletzt besucht am 8.12.2010, 19:34).

Verfügung stehen, kann anschliessend ermittelt werden, mit welchen Tastendrücken, Funktionsaufrufen oder Mausbewegungen bestimmte linguistische Einheiten bearbeitet werden und ob Autoren bestimmte Redigieroperationen immer mit der gleichen Folge von Tastendrücken ausführen – ob sie also eine bestimmte Sequenz von Funktionen trainiert haben und automatisiert ausführen – oder jeweils ad hoc eine solche Folge suchen und ausführen.

Daten, die in grossem Umfang eigentlich für die Progressionsanalyse aufgezeichnet wurden – hier interessieren nur die Veränderungen der Textoberflächen –, sollen in einem anderen Projekt daraufhin untersucht werden, ob sich statistische Muster finden lassen, aus denen sich Erkenntnisse und Empfehlungen für Schreibstrategien ableiten lassen, siehe [Perrin und Wildi 2009].

In beiden Projekten geht es also darum, Erkenntnisse über den Prozess zu gewinnen, die sich nicht allein aus der Beobachtung der Veränderung des Produktes ableiten lassen. Die Auswertung von Aufzeichnungen von Tastendrücken beschränkt sich auf die Beobachtung der Benutzer und eine Interpretation der Textveränderungen hinsichtlich der Redigierabsicht. Die Daten werden nicht daraufhin untersucht, wie Textbearbeitungsprogramme verändert werden können, welche Funktionen Autoren unterstützen oder behindern oder wie sich Anpassungen der Bedienoberflächen in Textbearbeitungsprogrammen auf den Schreibprozess auswirken.

Bei der Entwicklung von *EVE*, einem Editor für *VAX*-Computer mit dem Betriebssystem *VMS* in den 1980er Jahren, wurden *keystroke-logging* Daten von Benutzern anderer Editoren verwendet, um Designentscheidungen für Funktionen wie auch für die Anordnung von Funktionstasten auf der Tastatur zur treffen. Good [1985], Whiteside et al. [1982] und Rosson [1983] werten zu Beginn der 1980er Jahre solche Aufzeichnungen aus, um zu ermitteln, wie Benutzer Editoren tatsächlich verwenden und ob diese Aussagen sich mit gängigen Modellen zur Vorhersage von Verhalten decken. Hier geht es zudem um Empfehlungen zum Design von Editoren, um den Prozess des Schreibens und Bearbeitens von Dokumenten optimal zu unterstützen, siehe [Whiteside et al. 1982]. In den Arbeiten der Schreibforschung zur Analyse von *keystroke-logging*-Daten spielen diese Überlegungen jedoch keine Rolle und diese Arbeiten werden auch nicht referenziert.

2.4 Zusammenfassung

In diesem Kapitel haben wir Grundlagen der Schreibforschung dargestellt und auf das Redigieren als ein Element im Schreibprozess fokussiert. Innerhalb der Schreibforschung ist allgemein anerkannt, dass Schreiben ein Prozess ist und als solcher modelliert werden kann und muss. Schreibmodelle berücksichtigen alle Elemente, die den Schreibprozess beeinflussen können, und stellen einzelne Phasen als Subprozesse dar, die iterativ, rekursiv oder überlappend stattfinden. Solche Modelle können bislang jedoch nur verwendet werden, um einen bereits erfolgten Schreibprozess ex post zu erklären, sie können nicht als Vorhersagemodelle benutzt oder in Computerprogrammen implementiert werden.

Wir interessieren uns hauptsächlich für das Redigieren als eine Phase des Schreibprozesses. Das Verständnis dieses Subprozesses hat sich in den letz-

ten Dekaden gewandelt, wir bezeichnen heute alle Änderungen, die ein Autor zu beliebigen Zeitpunkten an einem Text vornimmt, als Redigieren – unabhängig davon, ob er bereits Geschriebenes verändert, seine Intention ändert, etwas korrigiert oder ergänzt. Redigieren umfasst nur zu einem geringen Teil die Korrektur von Fehlern, vielmehr geht es um die Veränderung bereits geschriebenen oder noch zu schreibenden Textes, um neue Ideen oder Argumente zu integrieren oder zu verdeutlichen.

Veränderungen eines Textes während des Schreibprozesses können mit Hilfe von Taxonomien klassifiziert werden. Klassen in solchen Taxonomien berücksichtigen bislang nicht die Intention des Autors und beziehen die verwendeten Werkzeuge und Funktionen nicht ein, sondern beschränken sich auf die Beobachtung der Textoberfläche zu verschiedenen Zeitpunkten.

Redigieroperationen bei der Arbeit mit Textbearbeitungsprogrammen können mit technischen Mitteln aufgezeichnet und ausgewertet werden. Bislang werden diese Aufzeichnungen lediglich hinsichtlich der Veränderung des Textes ausgewertet. Die Sicht auf das Redigieren am Computer fokussiert stark auf den Aspekt der Korrektur von Fehlern – obwohl die allgemein anerkannte Bedeutung von Redigieren sehr viel umfangreicher ist. Studien, die makrostrukturelles Redigieren am Computer untersuchen, existieren nicht.

Im folgenden Kapitel 3 betrachten wir den Stand der Technik von Schreibwerkzeugen. Dabei geht es einserseits um Textbearbeitungsprogramme – also Werkzeuge, mit denen tatsächlich geschrieben wird – und andererseits um Hilfsmittel, die den Autor bei der Bearbeitung eines Textes unterstützen sollen – etwa Nachbearbeitungsprogramme.

3 Stand der Technik

> *I also liked the function key that allowed one to delete the last word typed. I found that it was much faster to retype an entire word than to backspace the required number of letters, break into the thought process of spelling in the middle of the word, and then completing the word.*
>
> —E. David Callender, *An Evaluation of the AUGMENT System*, 1982

> *As we exchanged drafts for a conference presentation, we were entertained by some of the insane recommendations that our word processor's grammar and style checker suggested.*
>
> —Tim McGee, Patricia Ericsson, *The Politics of the Program: MS WORD as the Invisible Grammarian*, 2002

Im vorigen Kapitel 2 haben wir den Forschungsstand in der Schreibforschung beleuchtet und sind genauer auf das Redigieren eingegangen. In Abschnitt 1.4 haben wir argumentiert, dass typische Fehler in Texten durch ungenügende Unterstützung in Textbearbeitungsprogrammen erklärt werden können. Darum wenden wir uns in diesem Kapitel den Werkzeugen zu, die Autoren beim Erstellen und Bearbeiten von Texten mit dem Computer verwenden.

Computerprogramme zum Erstellen und Bearbeiten von Texten existieren seit den 1960er Jahren [siehe Haigh 2006]. Zunächst gehen wir in Abschnitt 3.1 auf Textbearbeitungsprogramme und deren Funktionen ein. Anschliessend stellen wir in Abschnitt 3.2 ausgewählte Projekte vor, die teilweise computerlinguistische Ressourcen und Methoden verwenden, um Autoren und Korrektoren beim Redigieren zu unterstützen. Die ersten dieser Programme, die das Nachbearbeiten von Texten allgemein oder das Anpassen eines Textes an bestimmte

Stilvorgaben unterstützen, wurden in den 1970er Jahren entwickelt [siehe Macdonald et al. 1982, Smithson 1986]. In Abschnitt 3.3 gehen wir speziell auf Prüf- und Korrekturprogramme ein. Projekte, die spezifische Phasen des Schreibprozesses unterstützen, für eine bestimmte Zielgruppe oder für eine bestimmte Textsorte entwickelt wurden, stellen wir in Abschnitt 3.4 vor. Wir können so zeigen, dass das Verständnis von Redigieren, das solchen Programmen zugrunde liegt, sich auf das Erkennen und Korrigieren von Fehlern beschränkt. Es gibt keine Funktionen, die Autoren bei der Modifikation ihrer Texte unterstützten und helfen typische Redigierfehler zu vermeiden.

3.1 Textbearbeitungsprogramme als Schreibwerkzeug

In diesem Abschnitt zeichnen wir zunächst die Entwicklung der Programme nach, die gemeinhin als Textbearbeitungsprogramme bezeichnet werden. Anschliessend gehen wir in Abschnitt 3.1.2 genauer auf die Funktionen ein, die heutige Textbearbeitungsprogramme für das Editieren von Texten zur Verfügung stellen.

3.1.1 Historischer Abriss

Der Begriff *word processing* wurde erstmals in den 1960er Jahren verwendet. Bezeichnet wurde damit nicht wie heute eine Softwareanwendung, sondern eine Komplettlösung aus Hardware und Software, der Marktführer war zu dieser Zeit *IBM* [vgl. Eisenberg 1992, Haigh 2006, Wohl 2006]. Vorhandene Technologien zum Speichern, Indizieren, Suchen und Finden von Texten wurden kombiniert und um Funktionen, wie sie vom Manipulieren von Programmen (also aus Texteditoren)[1] bekannt waren, erweitert. Zielpublikum waren Sekretärinnen, da diese in der Regel alle Schreibarbeiten erledigten. *Textbearbeitung* war also

> […] an ideal of centralizing typing and transcription in the hands of specialists equipped with technologies such as automatic typewriters. […] Quickly, however, the term acquired a more specialized meaning to refer almost exclusively to computerized text editing systems aimed at office applications. [Haigh 2006, S. 6]

Dabei ging es auch um vernetztes oder kollaboratives Arbeiten, Ziel war das «papierlose Büro» [Haigh 2006, S. 7]. Dieser Gedanke setzte sich jedoch erst durch und wurde kommerziell attraktiv, als Anfang der 1990er Jahre entsprechende Hardware auch für kleine Firmen oder gar Privatpersonen erschwinglich wurde. Zuvor – Mitte der 1970er Jahre – wurden Programme wie *Electric Pencil* oder *Easy Writer* von Hobbyprogrammierern für erste Microcomputer entwickelt, die jedoch nur einen kleinen Benutzerkreis hatten und sich später nicht gegen kommerzielle Produkte behaupten konnten [Bergin 2006a].

Neben Firmen arbeiteten auch Forschungseinrichtungen und Universitäten an der Entwicklung von Textbearbeitung und entsprechenden Editoren [Meyrowitz und van Dam 1982a, S. 336]. Diese setzten sich jedoch nicht durch, da Ende der 1980er, Anfang der 1990er Jahre kommerzielle Produkte sehr viel billiger

1. Zur Begriffsunterscheidung *Textbearbeitungsprogramm* und *Texteditor* siehe Abschnitt 1.1.

waren und auf PCs liefen, während die Programme der Forschungseinrichtungen Grossrechner benötigten [Vernon 2000, S. 332].

Die ersten kommerziell erfolgreichen Textbearbeitungsprogramme wie *WordStar* oder später *WordPerfect* verschwanden wieder, hauptsächlich durch Fehlentscheide des Managements der entsprechenden Firmen [vgl. Bergin 2006a,b, Eisenberg 1992]. *MS Word* für *Windows* wurde Mitte der 1990er Jahre nach *WordPerfect* der Marktführer im Bereich Textbearbeitungsprogramme und ist es nach den Verkaufszahlen bis heute [Bergin 2006b]. Welches Programm jeweils die meisten Benutzer hatte und sich am besten verkaufte, lag weniger in den Eigenschaften der Programme selbst als in Verkaufs- und Werbestrategien begründet – für *WordPerfect* stand beispielsweise eine kostenlose Telefon-Hotline zur Verfügung [Eisenberg 1992, Wohl 2006].

Teilweise wurden die Entwicklungen an Forschungseinrichtungen in kommerzielle Produkte übernommen, weil etwa beteiligte Forscher in die Industrie wechselten [Haigh 2006, S. 21], in der Regel war die Entwicklung jedoch nicht miteinander verknüpft. Die «alten» Textbearbeitungsprogramme wurden vergessen – weil sie auf aktueller Hardware nicht laufen, weil die Entwickler ein Institut verlassen etc. –, was bedauerlich ist, vergleicht man heute verfügbare Funktionen in Textbearbeitungsprogrammen mit den Möglichkeiten von Editoren aus den 1960er oder 1970er Jahren.

Als Beispiel betrachten wir *NLS* (oN-Line System), den von Douglas C. Engelbart 1968 vorgestellten Editor des Stanford Research Institutes.[2] *NLS* wurde in den 1960er Jahre von der Gruppe um Engelbart entwickelt. [Engelbart 1962] Ab 1978 wurden die kommerziellen Rechte an *Tymshare* übertragen, weil die Finanzierung durch die ARPA nicht weitergeführt wurde [Van Ittersum 2008, S. 149], und das Produkt erhielt den Namen *AUGMENT*.[3] *AUGMENT* wurde von *Tymshare* als «integrated office information system» vertrieben [Callender 1982, S. 29].

NLS arbeitete bereits 1968 mit verschiedenen Fenstern und konnte mit der Maus – einer weiteren Erfindung von Engelbart – bedient werden.

> NLS included many text editing capabilities of later word processors, including word wrap, search and replace, and scrolling, and the use of a mouse to select text to be cut and pasted between documents. Indeed Engelbart's system was much more complex than most of subsequent word processing systems, […] [Haigh 2006, S. 21]

NLS erlaubte zudem, die einzufügenden Elemente nach Zeichen, Wörtern oder umfassenderen Textelementen zu differenzieren [van Dam und Rice 1971, S. 110f]. Es ermöglichte sowohl das Schreiben von Texten als auch den Austausch von Daten, die Benutzung einer gemeinsamen Datenbank, E-Mail und das gleichzeitige Bearbeiten eines Dokuments (genannt *computer conferencing*) [Callender 1982, S. 32]. Die Benutzer konnten räumlich voneinander entfernt gleichzeitig gemeinsam an Dokumenten arbeiten.

2. Diese Vorstellung gilt als *mother of all demos*. Unter anderem wurde der Bildschirm des Computers von Engelbart live auf eine grosse Videoleinwand übertragen.
3. Siehe The Open Augment Consortium: «Open Augment», 2003, erhältlich im WWW: `http://www.openaugment.org/` (zuletzt besucht am 8.12.2010, 19:34).

Frühe Editoren, die für das Erfassen und Bearbeiten von natürlichsprachlichen Texten entwickelt wurden, hatten umfangreiche Editierfunktionen. Das *Hypertext Editing System* (HES) verfügte über die Möglichkeit, Ersetzungen über *regular-expression*-ähnliche Konstruktionen vorzunehmen [van Dam und Rice 1971, S. 108f]. *Easy Writer* erlaubte es, den Text am Bildschirm exakt so darzustellen, wie er später gedruckt wurde [Bergin 2006a, S. 36]. *WordStar* war das erste Textbearbeitungsprogramm, das als *What-you-see-is-what-you-get* (WYSIWYG) beworben wurde, siehe [Bergin 2006a, S. 39]. Ende der 1980er Jahre wurde erstmals die rote Unterkringelung von falsch geschriebenen Wörtern in *MS Word* verwendet [Bergin 2006b, S. 57]. Ausführliche Darstellungen der Geschichte der Textbearbeitungsprogramme finden sich bei Bergin [2006a,b], Eisenberg [1992] und Haigh [2006].

Im folgenden Abschnitt 3.1.2 gehen wir auf die in heutigen Textbearbeitungsprogrammen verfügbaren typischen Editierfunktionen ein. Neben einer allgemeinen Beschreibung weisen wir auf Schwierigkeiten hin, die sich für Autoren beim Verfassen oder Redigieren von natürlichsprachlichen Texten ergeben. Wir betrachten hier ausschliesslich Textbearbeitungsprogramme. In Abschnitt 4.1.1 widmen wir uns Funktionen in Texteditoren.

3.1.2 Editierfunktionen

Die grundlegenden Funktionen von Textbearbeitungsprogrammen sind in allen heutigen Programmen zu finden. Ein *Cursor* kann an beliebige Stellen platziert werden, um dort weitere Zeichen einzugeben oder verschiedene Funktionen auszuführen. In der Regel kann der Cursor mit der Maus positioniert werden. Alternativ kann man den Cursor mit den entsprechenden Tasten bewegen: nach rechts, nach links, nach oben und nach unten. Die Bewegungen nach oben und unten entsprechen dem Bewegen in vorhergehende und nachfolgende Zeilen, die Bewegungen nach rechts und links dem Bewegen zum nachfolgenden oder vorhergehenden Zeichen. Die Bewegung erfolgt also zeichen- und zeilenweise. Einige Textbearbeitungsprogramme bieten die Möglichkeit, den Cursor über Tastenkombinationen wortweise oder satzweise zu bewegen. *MS Word* erlaubt «wortweises» Bewegen des Cursors unter Verwendung der alt-Taste, was zusammen mit der shift-Taste auch für das «wortweise» Markieren verwendet werden kann.

Die Möglichkeiten der Bewegung des Cursors weisen schon darauf hin, wie die Editierfunktionen von Text-Editoren angelegt sind. Die grundlegenden Funktionen[4] `select` (auswählen/markieren), `cut` (ausschneiden), `copy` (kopieren), `paste` (einfügen)[5], `move` (bewegen), `merge` (zusammenfügen), `delete` (löschen), `search` (suchen), `search & replace` (suchen und ersetzen), `insert` (einfügen) und `undo` (rückgängig machen) operieren auf Zeichenebene. Der Benutzer kann eine bestimmte Anzahl Elemente auswählen und auf diese Auswahl dann die verschiedenen Kommandos anwenden. Die genannten Funktionen werden heute von Benutzern von Textbearbeitungsprogrammen als selbstverständlich vorausgesetzt.

4. Bezeichnet als «conventional word processor functions» [Piolat 1991, S. 262] oder «core operations» [Sharples und Pemberton 1990, S. 49].
5. Diese Funktionen hiessen nicht von Beginn an so, Wood und Allen et al. bezeichnen `copy-paste` (kopieren und einfügen) beispielsweise als `pick-put` [Wood 1981, S. 4], [Allen et al. 1981, S. 77].

Whiteside et al. [1982] untersuchten zu Beginn der 1980er Jahre, wie Editoren verwendet werden. Sie verglichen die Benutzung eines experimentellen Editors (*Editor Prototyping Tool (EPT)*) und eines kommerziellen Textbearbeitungsprogramms (*DEC WPS*). Während elf Tagen wurden alle Tastendrücke der Benutzer (sechs erfahrene «knowledge workers» und acht erfahrene Sekretärinnen), die ihre normale Arbeit erledigten, aufgezeichnet und anschliessend ausgewertet [Whiteside et al. 1982, S. 30]. Die Hälfte der *Arbeitszeit* mit jedem der Editoren wurde für Tastendrücke zur Zeicheneingabe verwendet; die Hälfte aller *Tastendrücke* war die Eingabe eines Zeichens. Ein Viertel der Zeit wurde für die Positionierung des Cursors benötigt und nur ein sehr kleiner Prozentsatz (7 % für EPT und 3,5 % für WPS) wurde für das Löschen von Zeichen aufgewandt [Whiteside et al. 1982, S. 32]. Aus bestimmten Abfolgen von Tastendrücken wurden so bestimmte Zustände ermittelt. So war das «Schreiben» – definiert als die ununterbrochene Folge von Tastendrücken zur Zeicheneingabe, heute als *burst* bezeichnet [Chenoweth und Hayes 2003, S. 103] – mehr als die Hälfte aller Tastendrücke und im Durchschnitt je «Schreibeinheit» etwa 17 Tastendrücke lang. Das Positionieren des Cursors war 34 % aller Tastendrücke und erforderte je Einheit 18,6 Tastendrücke. Gelöscht wurden hauptsächlich einzelne Zeichen. [Whiteside et al. 1982, S. 35f] Letzteres spricht dafür, dass lediglich Revisionen an der Oberfläche vorgenommen wurden, etwa das Korrigieren von Tippfehlern.

Rosson [1983] interessierte sich wenig später für ähnliche Fragen. Erfahrene Benutzer wurden beim Umgang mit *XEDIT* beobachtet, auch hier wurden die Tastendrücke aufgezeichnet. Je erfahrener ein Benutzer im Umgang mit dem Editor war, desto mehr Kommandos verwendete er. Allerdings liess sich kein persönliches Profil erstellen und keiner der Benutzer verwendete *alle* verfügbaren Kommandos des Editors [Rosson 1983, S. 172]. Rosson war überrascht, dass die Benutzer Kommandos nicht so verwendeten, wie sie von den Designern intendiert waren und effizienter gewesen wären. Er kommt aber zum Schluss, dass Editordesigner sich eher den Wünschen der Benutzer anpassen müssten, als darauf zu hoffen, dass sich die Benutzer dem Editor anpassen würden [Rosson 1983, S. 174].

Vergleichbare aktuelle Studien zur Verwendung der grundlegenden Funktionen in heutigen Textbearbeitungsprogrammen oder zur Verwendung elaborierter Funktionen existieren leider nur punktuell. Wir finden lediglich allgemeine Aussagen wie von Piolat: «Conventional word processor functions (cut, paste, delete, search, insert), even when used skillfully by writers do not seem to be sufficient for simultaneously planning and revising.» [Piolat 1991, S. 262]. Hausdorf [2005] kann zeigen, dass Benutzer eines neuen Textbearbeitungsprogramms (hier *ScientiFix*) Arbeitsprozesse, die sie aus anderen Programmen «gewohnt» sind, unreflektiert auf das neue Programm übertragen. Schon Norman [1983] benennt dieses Phänomen des Anwendens gewohnter Abläufe auf neue Werkzeuge als potenzielle Fehlerquelle.

In den folgenden Abschnitten präsentieren wir vier Beispiele zum Editieren von bereits geschriebenem Text – also zum Redigieren – unter Verwendung von Standardfunktionen in Textbearbeitungsprogrammen. Wir verwenden dazu jeweils die Funktionen, wie sie in *MS Word* vorhanden sind.

3.1.2.1 Editierbeispiel: Ausschneiden und Einfügen

Nur wenige Funktionen in Textbearbeitungsprogrammen operieren auf Strukturen, die nicht zeichenbasiert sind. Ein Beispiel ist die Verwendung der Funktionen für das Ausschneiden und Ersetzen von Zeichenketten in *MS Word* mit der Option `smart cut and paste`[6]. «smart» heisst hier, dass ein Arbeiten auf Wortebene erleichtert werden soll. Wird eine Zeichenkette markiert, die von Leerzeichen begrenzt ist, wird diese als «Wort» interpretiert. Wird diese Zeichenkette ausgeschnitten, müssen die umgebenden Leerzeichen reduziert werden. Abbildung 3.1 zeigt das Verhalten. Da es zusätzlich die Möglichkeit gibt, mit zweimaligem Drücken der Taste F8 oder durch Doppelklicken der linken Maustaste genau ein «Wort» (also eine Zeichenkette zwischen zwei Leerzeichen) auszuwählen, erleichtert dieses Verhalten das Editieren.

Abbildung 3.1: Verhalten von «smart cut and paste» in MS Word *beim Ausschneiden; «|» steht für die aktuelle Position des Cursors, der farbige Hintergrund steht für eine Markierung.*

1. Auswahl des auszuschneidenden Wortes	2. Resultat nach dem Ausschneiden von «*dem*»
`nach dem| Wort`	`nach |Wort`

Anders beim Einfügen einer Zeichenkette. Wird eine Zeichenkette eingefügt, werden automatisch jeweils ein Leerzeichen am Anfang und am Ende hinzugefügt. Für den Fall, dass das «Wort» *zwischen* zwei Wörtern eingefügt wird, sind die Leerzeichen nach Abschluss korrekt gesetzt. Für den Fall, dass eine Zeichenkette *in* ein Wort eingefügt wird, werden jedoch ebenfalls Leerzeichen vor und nach der Zeichenkette eingefügt, sodass das in Abbildung 3.2 gezeigte Schriftbild erscheint. In diesem Fall ist «smart» nicht smart genug, da nicht unterschieden wird, *wo*, d. h. in welchem Kontext, eine Zeichenkette eingefügt wird. Entsprechend müsste mit oder ohne Leerzeichen eingefügt werden. Es ist jedoch nicht möglich, diese Einstellung schnell zu ändern, der Benutzer wird sich hier also seinem Editor anpassen: Wird die Option `smart cut and paste` aktiviert, müssen regelmässig Leerzeichen entfernt werden, wird diese Einstellung nicht gewählt, müssen regelmässig Leerzeichen hinzugefügt werden.

Abbildung 3.2: Verhalten von «smart cut and paste» in MS Word *beim Einfügen.*

1. Cursor innerhalb eines Wortes	2. Resultat nach dem Einfügen von «*dem*»
`nach Wo|rt`	`nach Wo dem |rt`

3.1.2.2 Editierbeispiel: Vertauschen von Wortformen

Abbildung 3.3 auf der nächsten Seite zeigt, wieviele Schritte notwendig sind, um eine einzelne Redigieroperation auszuführen. Um die Reihenfolge der Konjunkte in «*lesen und schreiben*» zu vertauschen, sind mit einer Standardtextbearbeitungssoftware wie *MS Word* acht Schritte notwendig.[7] Wir gehen hier davon aus, dass lediglich Funktionen verwendet werden, die über Tastenkombinationen zu erreichen sind. Das Markieren von Zeichenketten kann auch durch optimales Platzieren der Maus, Klicken und Gedrückthalten der linken Maustaste während der Bewegung der Maus und anschliessendes Loslassen der Maustaste an der richtigen Stelle erzielt werden. Die Anzahl der notwendigen

6. In der deutschen Version heisst die Option `Ausschneiden und Einfügen mit Leerzeichenausgleich`.

7. In Abschnitt 7.1.2 zeigen wir, dass es möglich ist, mit weniger Schritten das gleiche Ergebnis zu erzielen, sofern der verwendete Editor weitere Funktionen anbietet, die auf dem Konzept *Wort* aufbauen.

Schritte verringert sich dadurch nicht, die kognitiven und motorischen Anforderungen sowie der zeitliche Aufwand steigen jedoch [vgl. Raskin 2000, S. 93ff]. Gleiches gilt für das Verwenden der `drag-and-drop`-Funktion in Word.

Ob ein Autor sich beim nächsten notwendigen Tauschprozess an diese optimale Sequenz erinnert, ist fraglich. Die Gefahr ist gross, dass einzelne Schritte vergessen werden, mehrfach ausgeführt werden oder in einer Reihenfolge, die zusätzliche Schritte erfordert. Vermutlich kann die Änderung schneller und mit weniger kognitivem Aufwand erfolgen, wenn die ursprüngliche Koordination gelöscht und komplett neu geschrieben wird.

Abbildung 3.3: Kürzestmögliche Sequenz von Schritten in MS Word*, um «lesen und schreiben» in «schreiben und lesen» zu ändern unter Verwendung von* `smart cut and paste` *(Mac Tastenkombinationen).*

Schritt	Kommando	Status
		`lesen und schreiben|`
1.	alt shift ←	`lesen und schreiben`
2.	Cmd X	`lesen und |`
3.	alt ←	`|lesen und`
4.	Cmd V	`schreiben |lesen und`
5.	alt shift →	`schreiben lesen und`
6.	Cmd X	`schreiben |und`
7.	alt →	`schreiben und |`
8.	Cmd V	`schreiben und lesen|`

Obwohl der Zugriff auf linguistische Einheiten (wie hier auf eine pragmatische Interpretation von *Wort*) möglich ist, existieren nur sehr wenige Funktionen, die diese verwenden. Funktionen, die auf grösseren linguistischen Einheiten operieren, wie Wortgruppen, Satzgliedern oder Sätzen, existieren nicht. Natürlich ist es möglich, eine der grundlegenden Funktion wie `delete` auf einen Satz anzuwenden. Der zu löschende Satz muss jedoch durch gezieltes Markieren mit der Maus oder Tasten ausgewählt werden, anschliessend kann er gelöscht werden. Es ist nicht möglich, den Cursor an eine beliebige Stelle innerhalb eines Satzes zu platzieren und dann etwa eine Funktion `aktuellen Satz löschen` auszuführen.

3.1.2.3 Editierbeispiel: Änderung der Gross-/Kleinschreibung

Der erste Buchstabe bestimmter Wortformen wird im Deutschen als Majuskel geschrieben, etwa für Namen und Substantive oder für das erste Wort eines Satzes. Während des Redigierens kann die Wortstellung im Satz so geändert werden, dass neu ein Wort das erste Wort des Satzes ist, das bisher klein geschrieben wurde. Es ist also notwendig, den ersten Buchstaben dieses Wortes zu kapitalisieren. Vermutlich muss das bislang erste Wort des Satzes neu kleingeschrieben werden. Hierfür muss zu dem betreffenden Buchstaben navigiert, dieser gelöscht und durch die entsprechende andere Version ersetzt werden.

Es existiert jedoch eine Funktion, die die Grossschreibung des ersten Buchstabens eines Wortes erlaubt – also eine Funktion, die auf der lingustischen Einheit *Wort* operiert. Um diese Funktion zu verwenden, ist es jedoch notwendig, eine Hand von der Tastatur zu lösen und mittels Maus im Menü an den Ort zu navigieren, von dem aus diese Funktion aufgerufen werden kann. Entsprechend *Hick's Law* und *Fitt's Law* [vgl. Raskin 2000, S. 93ff] erfordert dies jedoch viel

Zeit. Verwendet man nur die Tastatur, besteht die kürzestmögliche Sequenz aus sieben Schritten[8]:

1. Markieren des Wortes durch zweimaliges Drücken von F8
2. alt t (Aufruf des Menüs `Format`)
3. k (Auswahl von `Groß-/Kleinschreibung`)
4. w (Navigation zur Option `Ersten Buchstaben im Wort großschreiben`, diese Situation zeigt Abbildung 3.4)
5. Tabulatortaste, um zu `OK` zu navigieren
6. ↵ (Bestätigen)

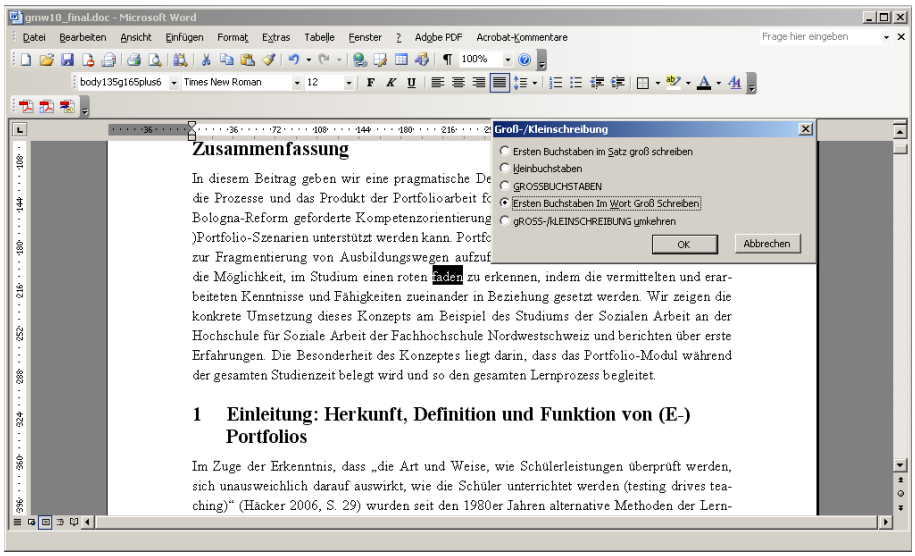


Abbildung 3.4: Funktion zur Änderung der Gross-/Kleinschreibung eines Wortes in MS Word.

3.1.2.4 Editierbeispiel: Ersetzen von Wörtern

Betrachten wir als weiteres Beispiel die Funktion `search & replace` (suchen und ersetzen). Beim Vertauschen von Konjunkten, wie bereits in Abschnitt 3.1.2.2 gezeigt, wird eine konkrete Textstelle bearbeitet, die der Autor bewusst auswählt und vor dem Aufrufen bzw. Ausführen der beteiligten Funktionen im Bildschirmausschnitt sieht. Dies ist beim Suchen und Ersetzen in der Regel nicht der Fall. Weder weiss der Autor, wieviele Textstellen betroffen sind, noch sind diese Textstellen beim Aufruf der Funktion auf dem Bildschirm sichtbar. Eventuell ist *eine* solche Textstelle sichtbar. Es ist jedoch auch möglich, dass der Autor sich zum Ersetzen eines Wortes oder einer Wortgruppe entschliesst, *ohne* ein entsprechendes Vorkommen auf dem Bildschirm zu sehen.

Nach dem Aufruf der Funktion – über eine Tastenkombination oder einen Menüeintrag – definiert der Autor die global zu ersetzenden Zeichen und die Zeichen, die stattdessen verwendet werden sollen. Wir benutzen hier bewusst den Begriff *Zeichen*, da die Konzepte *Wort* oder *Wortgruppe* in Textbearbeitungsprogrammen für `search & replace` nicht verwendet werden. Entsprechend können Fundstellen der zu ersetzenden Zeichen sowohl ganze Wortformen als

8. Auch hierfür zeigen wir in Abschnitt 7.1.2, dass kürzere Sequenzen möglich sind.

auch Wortanfänge, Wortenden, Zeichenketten innerhalb eines Wortes oder über Wortgrenzen hinweg sein. Möchte ein Autor tatsächlich alle Vorkommen eines *Wortes* durch die entsprechenden Wortformen eines anderen Wortes ersetzen – soll etwa konsistent statt «*Zelt*» «*Hütte*» verwendet werden –, ist dies ein langwieriger Prozess.

Für das zu ersetzende Wort müssen zunächst alle Wortformen des Paradigmas gesucht werden. Da im Deutschen verschiedene Kasus – im Fall von Substantiven –, verschiedene Genus – im Fall von Adjektiven – und verschiedene Tempora – im Fall von Verben – in der Regel in der Oberfläche eines Wortes markiert werden, hat ein Wort der flektierenden Wortklassen mehrere Wortformen. Die Anzahl der Wortformenoberflächen je Wort variiert jedoch, d. h., nicht alle Substantive haben die gleiche Anzahl verschiedener Wortformoberflächen. Damit ergibt sich eine zweite Schwierigkeit: Wird eine bestimmte Oberfläche einer Wortform gefunden, muss die Kategorie[9] der Wortform ermittelt werden, um sie durch die korrekte Wortform des Ersatzwortes ersetzen zu können. Tabelle 3.5 zeigt am Beispiel der Wörter «*Zelt*», «*Haus*» und «*Hütte*» die unterschiedliche Anzahl von verschiedenen Wortformoberflächen für Wörter und die unterschiedlichen Kategorien für eine Wortformoberfläche. Auch wenn zu ersetzendes Wort und Ersatzwort die gleiche Anzahl Wortformen aufweisen, können diese jeweils unterschiedlichen Kategorien angehören.

Abbildung 3.5: Paradigmen von «Zelt», «Haus» und «Hütte» in distinktiver Kategorisierung.[10]

Wort	Wortform	Kategorien
Zelt (*n*, (e)s/e)	Zelt	`Nom Sg, Dat Sg, Acc Sg`
	Zeltes	`Gen Sg`
	Zelts	`GenSg`
	Zelte	`Dat Sg, Nom Pl, Gen Pl, Acc Pl`
	Zelten	`Dat Pl`
Haus (*n*, (e)s/er)	Haus	`Nom Sg, Dat Sg, Acc Sg`
	Hauses	`Gen Sg`
	Hause	`Dat Sg`
	Häuser	`Nom Pl, Gen Pl, Acc Pl`
	Häusern	`Dat Pl`
Hütte (*f*, -/en)	Hütte	`Nom Sg, Gen Sg, Dat Sg, Acc Sg`
	Hütten	`Nom Pl, Gen Pl, Dat Pl, Acc Pl`

Die scheinbar einfache Redigieranweisung «Ersetze im gesamten Text «*Zelt*» durch «*Hütte*»!» mündet in eine komplexe Abfolge von Handlungen und Überlegungen, die vom Autor sehr viel Aufmerksamkeit erfordert. Die von Taylor so genannte «brain-to-hand-to-keyboard-to-screen connection» [Taylor 1987, S. 79] beansprucht also sehr viele Ressourcen, die damit für eigentliche inhaltliche Überlegungen nicht mehr zur Verfügung stehen: *Warum* soll diese Ersetzung vorgenommen werden? Ist sie wirklich sinnvoll? Sollte sie rückgängig gemacht werden? Ist ein anderes Ersatzwort vielleicht noch besser geeignet? Werden diese Überlegungen *während* des Umsetzens der Ersetzungsoperation gemacht und führen zu einer Abwandlung – z. B. «*Zelt*» sollte besser durch «*Haus*» und nicht durch «*Hütte*» ersetzt werden –, gestaltet sich das Ganze

9. Wir bezeichnen mit Kategorie die Wortart und die morphosyntaktischen Eigenschaften einer Wortform.

10. Zum Unterschied zwischen distinktiver und exhaustiver Kategorisierung siehe [Hausser 2001, S. 244 und S. 346ff]. Für die Einträge in der Spalte **Kategorien** gilt, dass jede Wortform ein Substantiv ist, diese Information wird in der Tabelle aus Platzgründen nicht dargestellt.

noch schwieriger: Die bisher ausgeführten Änderungen müssen rückgängig gemacht bzw. müssen die entsprechenden Textstellen anders als die noch nicht editierten behandelt werden.

Diese Bearbeitungen beanspruchen viele kognitive Ressourcen des Autors, da sehr viele Funktionen in einer bestimmten Reihenfolge ausgeführt werden müssen, um das gewünschte Resultat zu erzielen. Solche langen und komplexen Sequenzen sind inhärent fehleranfällig [siehe Norman 1983, S. 257]. Das Vermeiden von langen Sequenzen von Funktionen, um ein bestimmtes Ziel zu erreichen, verhindert das Auftreten von Unterbrüchen, die zu Fehlern führen können.

Lokale Ersetzungen von Wörtern, die für das neue Wort Buchstaben des zu ersetzenden Wortes verwenden, wie in den Abbildungen 3.6 und 3.7 gezeigt, nutzen explizit den Fakt aus, dass das Konzept des Wortes in MS *Word* nicht zur Verfügung steht. In beiden Fällen wird jeweils der erste Buchstabe des ursprünglichen Wortes (bzw. der ursprünglichen Wortgruppe) behalten und für den ersten Buchstaben des neuen Wortes benutzt. Beobachtet man lediglich die Veränderung an der Textoberfläche, wird also nicht «*proverb or idiom*» durch «*phraseme*» ersetzt, sondern «*roverb or idiom*» durch «*hraseme*» (Abbildung 3.6).

Abbildung 3.6: Ersetzen von «proverb or idiom» durch «phraseme» in MS Word *(Textausschnitt).*

Statt «*Deutsches Seminar*» durch «*Department of German*» wird «*eutsches Seminar*» durch «*epartment of German*» ersetzt (Abbildung 3.6).

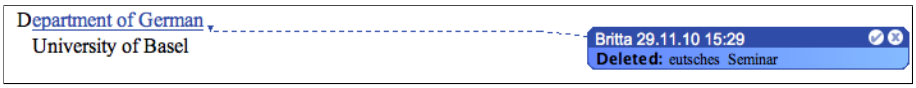


Abbildung 3.7: Ersetzen von «Deutsches Seminar» durch «Department of German» in MS Word *(Textausschnitt).*

Zusammenfassend lässt sich feststellen: Die grundlegenden Funktionen in Textbearbeitungsprogrammen sind zeichenbasiert. Editier- und Redigieranweisungen sind dagegen in der Regel Operationen, die sich auf linguistische Einheiten beziehen:

> […] the writer is defining and operating upon a unit of text larger than the letter. He is working topically—thinking and writing in terms of verbal units or topics, whose meaning transcends their constituent letters or words. […] We conceive of our text as a set of verbal gestures, large and small. To write is to do things with topics—to add, to delete, and rearrange them. [Bolter 1989, S. 131]

Diese Operationen müssen jedoch mit zeichenbasierten Funktionen umgesetzt werden. Dieser Widerspruch erzwingt die Umwandlung von beabsichtigten Operationen, die auf linguistischen Einheiten basieren, in lange, komplexe Sequenzen zeichenbasierter Funktionen, die Autoren kognitiv stark beanspruchen, deren Ausführung relativ lange dauert und die zu typischen Fehlern in Texten führen.

3.2 Projekte zur automatisierten Schreibunterstützung

In diesem Abschnitt stellen wir Projekte vor, die Autoren, Korrektoren oder Lektoren Funktionen zur Bearbeitung von Texten anbieten. Wir können grob zwei Gruppen unterscheiden: Systeme und Funktionen, die für die *Nachbearbeitung* von Texten (engl. *post-writing*) eingesetzt werden, und Systeme und Funktionen, die für den interaktiven Einsatz *während* des Schreibens konzipiert wurden. Beide Varianten sind hauptsächlich auf die Korrektur von Fehlern ausgerichtet, nicht auf die Bereitstellung von Funktionen zur tatsächlichen Bearbeitung des Textes. Sie bieten damit nur für einen Teilaspekt des Redigierens Unterstützung.

Nachbearbeitungssysteme werden erst nach einer Schreibphase eingesetzt, wie etwa Hull et al. [1987] darstellen. Sie werden also in einer expliziten und bewussten Redigierphase verwendet, [Sharples 1999, S. 105ff] nennt dies «revising as seperate activity». Anfangs wurden Programme, die einen Text, der einen bestimmten Grad an Vollständigkeit erreicht hat, hinsichtlich verschiedener Aspekte automatisch prüfen, als eigenständige Programme eingesetzt und waren nicht in das Textbearbeitungsprogramm integriert. Solche Systeme liefern Kommentare zu fehlerhaften oder problematischen Textpassagen in einer getrennten Datei oder einem eigenen Fenster; der eigentliche Text muss dann in der Regel von Hand überarbeitet werden. Wird der Kommentar in einer anderen Datei geliefert, kommt noch die Suche nach der entsprechenden Textstelle hinzu, anschliessend muss der Kommentar zur Fundstelle in eine Lösung umgesetzt werden. [Holt 1989, S. 56f] Dies ist sehr ähnlich der Verwendung eines Ausdrucks auf Papier zum Korrekturlesen und entspricht nicht dem Redigieren, wie es Autoren während des Schreibens tun.

In den folgenden Abschnitten stellen wir die wichtigsten Programme und Projekte vor. Das Zielpublikum dieser Programme ist unterschiedlich: Einige, wie *Writer's Workbench*, sollten vom Autor selbst verwendet werden. Andere, wie *Editor's Assistant*, waren für Korrektoren (engl. *copy-editors*) gedacht, also für die Personen in Verlagen oder Publikationsinstitutionen, die auf Konsistenz der von den Autoren eingesandten Texte mit Stil- und Orthographieregeln des Verlegers achten und entsprechende Korrekturen vornehmen.

3.2.1 *The Writing Workshop*

The Writing Workshop, entwickelt von der *Milliken Publishing Company*[11] für Englisch, ist neben *Writer's Workbench* (siehe Abschnitt 3.2.2) eine der «grossen alten» Schreibhilfen, bestehend aus verschiedenen Programmen zur Vor- und Nachbereitung des Geschriebenen (engl. *pre-writing* und *post-writing*) sowie einem Textbearbeitungsprogramm (dem *Milliken Word Processor*). Für die Nachbearbeitung stehen verschiedene Funktionen zur Verfügung: Prüfung der Orthographie (*spelling checker*), Prüfung der Einhaltung einfacher Stil- und Grammatikregeln (*mechanics checker*) und Korrekturlesen (*proofreader*). Die Stil- und Grammatikprüfung untersucht mit einfachen Regeln die Verwendung von Pronomen, Konjunktionen und von mit bestimmten Wörtern beginnenden Wortgruppen. Zusätzlich werden Homophone markiert und Interpunktion

11. http://www.millikenpub.com/ (zuletzt besucht am 8.12.2010, 19:34), ursprünglich St. Louis, MO, USA, wurde 2008 von der *Lorenz Corporation* aufgekauft.

überprüft. Für eine vollständige Beschreibung der Funktionalität verweisen wir auf [Smithson 1986, S. 84ff]. Smithson betont, dass lediglich Vorschläge unterbreitet werden – nicht alle Funktionen arbeiten zufriedenstellend:

> One of the few grammar rules that has no exceptions in American English is that periods and commas go inside quotation marks, but students will be challenged by "Typing Checker" every time they follow this rule. [Smithson 1986, S. 87]

Diese Funktionen ähneln den heute bekannten Stil- und Grammatikprüfern, wie wir sie in Abschnitt 3.3 beschreiben: Es werden diejenigen Elemente gekennzeichnet, die vom System als fehlerhaft klassifiziert wurden. Anders die Korrekturlesekomponente des *Writing Workshop*: Hier wird jeder einzelne Satz hervorgehoben und der Autor muss die Korrektheit selbst beurteilen [siehe Smithson 1986, S. 88ff]. Es wird also die Konzentration auf ein einzelnes Element – einen Satz – gefördert, das der Autor selbst auf Korrektheit überprüft und eventuell ändert.

3.2.2 *Writer's Workbench*

Writer's Workbench ist eines der wenigen Programme, die nach wie vor erhältlich sind, weiter gepflegt werden und – obwohl auch *Writer's Workbench* ursprünglich ein eigenständiges Programm war – in heutige Textbearbeitungsprogramme integriert werden können. Es wurde Ende der 1970er Jahre von *Bell Labs* für Englisch entwickelt und lief anfangs nur unter dem Betriebssystem *UNIX*. Heute ist es für verschiedene Betriebssysteme erhältlich. Die Aufgabe des Programms ist die Analyse von Dokumenten hinsichtlich Orthographie- und Grammatikregeln und das Vorschlagen von Verbesserungen, wenn das Dokument diesen Regeln nicht entspricht. Geprüft werden auch Regeln, die eher dem Bereich «Stil» zuzuordnen sind, wie die Verwendung des Passivs. Siehe [Day 1988, Macdonald et al. 1982] oder [Stutz 2007] zur Funktionsweise.

Neben vielen sehr vorteilhaften Funktionen, wie der Möglichkeit, selbst einen Standard auf Grund von eigenen Texten zu definieren [Day 1988, S. 65f] oder die Orthographieregeln zu ergänzen und ein eigenes Lexikon anzulegen (etwa um Abkürzungen korrekt zu erkennen) [Macdonald et al. 1982, S. 106f], weist *Writer's Workbench* durchaus Schwächen auf. Laut [Day 1988, S. 65ff] seien die Rückmeldungen des Systems, bezogen auf das Korrekturlesen, oft nicht nützlich und die vorgeschlagenen Änderungen ungeeignet, böten jedoch Anlass, über das Geschriebene nachzudenken und somit selbst zu einer Lösung zu gelangen.

Die Argumentation ähnelt sehr derjenigen in Bezug auf *The Writing Workshop*: Die Ergebnisse der Analysen sind nicht perfekt, sondern sollten als Hilfsmittel verwendet werden, um den Text selbständig zu beurteilen: «[…] finding a match in text doesn't mean that the word or phrase is necessarily bad […] Sometimes, the suggestions are just hints or warnings that you can safely ignore when you understand what they're telling you.» [Stutz 2007, S. 21].

Die optisch sehr viel ansprechenderen aktuellen Werkzeuge verführen Autoren dazu, die Rückmeldungen der Systeme für endgültig und allein gültig anzusehen und die vorgeschlagenen Änderungen bedenkenlos umzusetzen. Dies ist umso erstaunlicher, wenn man bedenkt, dass sich die Grundlagen und verwendeten

Regeln dieser Systeme seit *Writer's Workbench* nicht wesentlich verändert oder gar verbessert haben [vgl. Dale und Douglas 1996, S. 127]. *Writer's Workbench* wird weiterhin entwickelt und beispielsweise als Ergänzung zu *MS Word* verkauft. Abgesehen von Modifikationen der Bedienoberfläche, sind keine wesentlichen Änderungen festzustellen.[12] Die Firma *EMO Solutions, Inc.* hat die Originalprogramme von *AT&T*, dem Nachfolger von *Bell Labs*, 1991 gekauft und 1999 für Windows angepasst.[13]

3.2.3 *RUSKIN*

Williams [1990a] nennt verschiedene Probleme von Programmen zur Nachbearbeitung von Texten, die sich auf Schreibsoftware generell übertragen lassen: Die tatsächlichen Bedürfnisse der Autoren würden nicht berücksichtigt, die Oberflächen der Programme seien schlecht, Analysen von Prüf- und Korrekturprogrammen seien unangemessen oder sogar falsch, alles in allem seien es lediglich Umsetzungen schlechter ad hoc-Ansätze [Williams 1990a, S. 3].

Das Projekt «Specifying User Writing Needs in the Automated Office» sollte untersuchen, welche Bedürfnisse Autoren von englischen Texten haben und welche Unterstützung der Computer bieten kann. Einerseits sollten also Bedürfnisse von Autoren berücksichtigt werden, andererseits wurden Funktionen und Möglichkeiten von Software evaluiert. Dabei war von vornherein klar, dass ein Produkt zur *Nachbearbeitung* von Texten implementiert werden sollte: *RUSKIN* [Williams 1990b, S. 5f]. *RUSKIN* wurde in *PROLOG* für *IBM* PCs entwickelt [Williams 1989, S. 4ff].

Benutzerbedürfnisse können entsprechend Williams [1989, S. 2f] klassifiziert werden als: (a) eigenen Text verbessern, (b) fremden Text verbessern, (c) Text entsprechend vorgegebenen Spezifikationen schreiben, (d) Schreiben lernen und (e) Schreiben lehren. *RUSKIN* sollte Unterstützung für die beiden letzten Punkte liefern, also für das Lehren und Lernen des Schreibens von Texten. Die Zielgruppe waren Techniker, Studenten technischer Fächer und Autoren technischer Dokumente. *Writer's Workbench* wurde für diese Ausrichtung als nicht geeignet angesehen, da es nicht flexibel zu verwenden war, dem Benutzer nicht die maximale Kontrolle über alle Funktionen überliess und vor allem keine attraktive Bedienoberfläche hatte. Zudem konnten mit *Writer's Workbench* nur Überprüfungen auf Wortebene durchgeführt werden; *RUSKIN* verwendete syntaktische Analyse[14].

Die Spezifikation basierte jedoch nicht auf Befragungen oder Beobachtungen von Autoren innerhalb der Zielgruppe, sondern lediglich auf Selbstbeobachtung der Beteiligten des Projekts. So wurde erst gegen Ende des Projekts festgestellt, dass Autoren eher Unterstützung *während* des Schreibprozesses wünschten und nicht in einer expliziten Nachbearbeitungsphase: «[…] the concept of postwriting software implies a linear model of the human writing process which is at best simplistic and at worst may be completely misleading.» [Williams 1989, S. 6]. Bereits existierende Programme zur Nachbearbeitung von Texten

12. Siehe «Writer's Workbench», 6.8.2010, im WWW `http://www.writersworkbench.com` (zuletzt besucht am 8.12.2010, 19:34).
13. Persönliche Kommunikation mit Greg Oij, E-Mail vom 4.11.2008, Message-ID `<RE5ROTMwVSpLTC1TSFFBNjkzNjI4ODkw@GOVISTA>`.
14. Allerdings lieferte der probabilistische Parser lediglich zu 75 % korrekte Analysen [Williams 1993, S. 117].

waren den Autoren der Zielgruppe grossenteils unbekannt oder wurden als «schwer zu bedienen» eingestuft [Williams 1990b, S. 7].

RUSKIN wurde klar als Nachbearbeitungs-System konzipiert [Williams 1989, S. 1], eine interaktive Benutzung war nicht vorgesehen. Aus der Kombination von Kontextvariablen (wie Merkmalen des Publikums, Ziel und Inhalt des Textes), Daten zum Autor und Analyseergebnissen von Textelementen (hinsichtlich ihrer Länge, bestimmter Wortlisten, syntaktischer Merkmale und typographischer Eigenschaften) wurden generelle Aussagen über den Text gewonnen. Diese Rückmeldungen sollten Autoren ermöglichen, ihren Text in bestimmten Punkten (etwa die Satzlänge in Wörtern) auf bestimmte Kontextvariablen hin (etwa das Bildungsniveau des Publikums) anzupassen, siehe [Holt et al. 1990, S. 27] und [Williams 1989, S. 7f].

Jedoch war es schwierig, neue Regeln zu implementieren oder vorhandene Regeln anzupassen; man benötigte jeweils einen Programmierer. Es wurde zwar ein interaktives Werkzeug – *LINK* – zur Entwicklung von Regeln implementiert, dies löste das Problem jedoch nicht. *LINK* diente zum Entwickeln und Testen von Regeln und war implementiert in *KEE* (Knowledge Engineering Environment), basierend auf *Lisp*. [Holt et al. 1990, S. 27ff] Die Kontextvariablen mussten von jedem Benutzer für jeden Text erfasst werden. Es konnte jedoch festgestellt werden, dass diese Notwendigkeit den meisten Autoren bereits half, ihren Schreibprozess zu optimieren, da sie zu Reflexionen über den Text angehalten wurden [siehe Williams 1989, S. 12].

McGowan [1992] argumentiert, dass die *Interaktion* des Autors mit dem System den grössten Gewinn für den Autor liefere. Die Erstellung von Stilanalysen wie von *RUSKIN* liesse den Autor jedoch mit der Interpretation allein [McGowan 1992, S. 298f]. McGowan entwickelt darum *McRuskin*, das mehr Interaktion mit dem Autor und direktes Manipulieren des Textes während des Überprüfens erlaubt, aber weiterhin ein Nachbearbeitungsprogramm ist.

3.2.4 *Writer's Assistant*

Writer's Assistant war ein grossangelegtes Projekt Ende der 1980er, Anfang der 1990er Jahre, finanziert von *British Telecom* [Sharples und Pemberton 1990, S. 50], ebenfalls für Englisch. Es sollte ein Werkzeug für Autoren entwickelt werden, das klar auf den Bedürfnissen der Autoren basierte und nicht auf den technischen Möglichkeiten der damaligen Zeit. Der Ausgangspunkt war also vergleichbar mit dem für *RUSKIN*. Die Anforderungen bezogen sich auf Schreibansätze, Schreibstrategien und Schreibtechniken. Zusätzlich sollten sich – als informatische Anforderung – die grundlegenden Funktionen jeweils identisch bedienen lassen und sich identisch verhalten. Einem Autor sollte also ein möglichst niedriger Lernaufwand zugemutet werden:

> The general core operations such as *cut*, *paste*, *move*, *copy*, *merge* and *undo* should be performed in a similar way regardless of context. This greatly cuts down the time to learn a new application, as the user can infer the existence of operations and how to perform them from previous experience. [Sharples und Pemberton 1990, S. 49]

Sharples et al. [1989] und Sharples und Pemberton [1990] berichten von einem Prototypen, der auf grosse Begeisterung der Versuchspersonen stiess, jedoch wurde der *Writer's Assistant* im geplanten Umfang nie umgesetzt. Geplant war *Writer's Assistant* als System, dessen Fokus auf den Aktivitäten von Autoren liegen sollte, also deutlich über eine lediglich Ergänzung zu bestehenden Textbearbeitungsprogrammen hinausging: «The Writer's Assistant is a computer-based *cognitive support system* for people who create complex documents as part of their professional live.» [Sharples et al. 1989, S. 22][15]. Als Programmiersprache wurde *Poplog*[16] verwendet.

Geplant war eine Kombination aus Textbearbeitungsprogramm, *ideas processor* und *outliner/structure editor*, die eine vollständige Unterstützung des Schreibprozesses ermöglicht und verschiedene Sichten auf den entstehenden Text erlaubt hätte [Sharples et al. 1989, S. 22]. Als Ausgangspunkt diente das Schreibmodell von Flower und Hayes [1981], jedoch wurde schnell festgestellt, dass dieses Modell sich kaum für die Implementierung eignet [Sharples et al. 1989, S. 26], wie Hayes [2001] selbst konstatiert.

3.2.5 *Editor's Assistant*

Ein weiteres grossangelegtes Projekt aus dem gleichen Zeitraum wie *Writer's Assistant* war *Editor's Assistant*, ebenfalls für Englisch. Hier ging es nicht um ein umfassendes Werkzeug zur Unterstützung des gesamten Schreibprozesses, sondern explizit um Werkzeuge zur Unterstützung des *Redigierens*.

Das Konzept beruhte auf zwei Annahmen: (a) Empfehlungen oder Richtlinien zu Grammatik, Stil und Interpunktion können als Regeln formuliert und in einer Wissensbasis abgelegt werden, (b) diese Regeln können auf einen Text angewandt werden, vgl. [Dale 1990, S. 59] und [Dale und Douglas 1996, S. 128ff]. Die Regeln ergeben sich aus den Funktionsprinzipien einer Sprache (bezogen auf Grammatik) und aus Vorgaben für die Textgestaltung von Verlagen o. Ä. (ein sogenannter Haus-Stil). Letztere beziehen sich vor allem auf Elemente der Typographie oder Einheitlichkeit von Terminologie und Darstellung von Fakten [Dale 1990, S. 59]. Redigieren wird hier also auch im weitesten Sinne als *Fehlerkorrektur* betrachtet, nicht als kreativer Akt der Bearbeitung und Veränderung von Text.

Ein Korrektor oder Lektor (engl. *copy-editor*)[17] arbeitet auf mehreren Ebenen: Struktur und Organisation eines Textes werden verändert, der Text wird an einen Haus-Stil angepasst und anschliessend Korrektur gelesen [Dale 1989, S. 16]. Nicht alle Aufgaben können und sollen automatisiert werden. Diejenigen Arbeiten, die sich jedoch an den Computer «delegieren» liessen, entlasteten den Korrektor und ermöglichten es ihm, sich auf anspruchsvollere Aufgaben zu konzentrieren [Dale 1990, S. 60].

Die Regeln und ihre Anwendung auf den Text beruhen einerseits auf Musterabgleichen (engl. *pattern matching*) – wenn es etwa darum geht sicherzustellen, dass Datumsangaben in einem einheitlichen Format erfolgen, – andererseits werden Funktionen aufgrund von Schlüsselwörtern angewandt – so können

15. Hervorhebung im Original.
16. `http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html` (zuletzt besucht am 8.12.2010, 19:34).
17. Wir verwenden für *copy-editor* im Folgenden die deutsche Bezeichnung *Korrektor*.

beispielsweise Kilometerangaben in Meilen umgerechnet oder Abkürzungen aufgelöst werden. Dale [1990, S. 62], Dale [1989, S. 17ff] und Dale und Douglas [1996, S. 131ff] erläutern Beispielfunktionen.

Der Text wird jeweils vollständig geparst, die Regeln werden nacheinander auf ihn angewandt. Der *Editor's Assistant* wird also in einem expliziten Nachbearbeitungsschritt eingesetzt – es liegt eine vollständige Textfassung vor, die geprüft und korrigiert wird. Die Stellen, an denen eine Regelverletzung festgestellt wird, werden hervorgehoben. [Dale 1989, S. 18] Das System bietet dem Korrektor verschiedene Optionen zur Behandlung einer Fundstelle, d. h. einer Regelverletzung: (a) Die Fundstelle kann durch etwas anderes ersetzt werden, (b) diese Ersetzung kann auch auf alle weiteren Fundstellen (d. h. Verletzungen derselben Regel) angewandt werden oder (c) die Regel, deren Verletzung zu dieser Fundstelle führte, kann an dieser Fundstelle oder für den gesamten Text deaktiviert werden, siehe [Dale 1989, S. 18] und [Dale 1990, S. 61f].

Erste Versuche mit professionellen Korrektoren zeigten klar, dass eine vollautomatische Prozedur nicht möglich und nicht erwünscht war: «First, from consultations we have carried out with working editors, it has become obvious that we need to allow the user to edit the text manually, simultaneously with the system's normal proofing operation.» [Dale 1990, S. 66] Die Korrektoren fühlten sich zu sehr eingeschränkt.

Später legen Dale und Douglas [1996, S. 128] dar, dass das vorgeschlagene Konzept zum Redigieren auch für andere als die ursprünglich intendierten Ziele eingesetzt werden könne. Eine mögliche Anwendung sei das Erkennen von Syntaxfehlern. Darunter fallen die Verletzung von Bedingungen (z. B. Subjekt-Verb-Kongruenz), die lexikalische Verwechslung von Homophonen, schwer verständliche Syntax, d. h. zu lange, zu sehr geschachtelte Sätze, das Aufspüren von fehlenden oder überflüssigen Elementen, wie Kommas oder Relativpronomen. Sie schränken jedoch ein, dass die meisten Fehler nicht ohne Analyse von Syntax und Pragmatik zu beheben seien. [Dale und Douglas 1996, S. 134f] Die Verbindung von computerlinguistischen Ressourcen und Textbearbeitung nennt Dale «intelligent text processing» [Dale 1989, S. 8]. Auch hier handelt es sich jedoch lediglich um das Finden, Diagnostizieren und Korrigieren von *Fehlern* in einer *abschliessenden* expliziten Schreibphase, nicht um das Redigieren *innerhalb* des Schreibprozesses zur Entwicklung des Textes.

Die verwendeten Systeme zur syntaktischen Analyse müssten entsprechend robust sein [Dale und Douglas 1996, S. 136ff]. Dies zu implementieren sei jedoch schwierig:

> An unfortunate consequence of broad grammatical coverage is that many sentences which appear unambiguous to the human reader are given multiple analyses by the computational grammar, and this makes it difficult to apply the kinds of techniques discussed here in a way that is not frustrating for the user. [Dale und Douglas 1996, S. 142f]

Das System wurde nach der Implementierung einiger beispielhaften Funktionen aus finanziellen Gründen nicht weiterentwickelt und kam nie auf den Markt.[18]

18. Persönliche Information von Robert Dale, 1. Juni 2008.

3.2.6 *Intelligent Workstation*

Intelligent Workstation wurde von Kempen et al. [1986] als eine Instantiierung der «fünften Generation von Textbearbeitungsprogrammen» [Kempen et al. 1986, S. 372] (bezeichnet als *Author Environment* und *Author System*) konzipiert. Entwickelt wurde es in einem von der EU geförderten Projekt. Ausgangspunkt war diese Beobachtung: «A major drawback of current word processors and text editors is their almost complete ignorance of language structure. What linguistic knowledge they have is restricted to spelling and hyphenation.» [Kempen et al. 1986, S. 365] Es sollte versucht werden, verschiedene Sichtweisen auf Text und verschiedene Aspekte der Bearbeitung von niederländischem[19] Text in einem Werkzeug zu vereinen: rhetorische, linguistische, typographische und graphemische Funktionen sollten implementiert werden. Der Prototyp wurde in *Common Lisp* implementiert und lehnte sich an die Funktionalitäten von *Emacs* an. [Kempen et al. 1986]

Linguistisch unterstützte Funktionen in *Intelligent Workstation* basieren auf der vollständigen syntaktischen Analyse von Sätzen und deren Repräsentation in Parse-Bäumen. Der Autor kann diese Bäume manipulieren, anschliessend wird die sprachliche Entsprechung eines geänderten Baums zurück in den Text geschrieben. [Kempen et al. 1986, S. 368f und S. 371] Dies ist eine sehr genaue Entsprechung von *syntax-orientierten* Funktionen, wie wir sie für Texteditoren in Abschnitt 4.1.1 zeigen. Der Autor wird gezwungen, vollständige Sätze zu schreiben, bevor er sie redigieren kann. Wie wir in Abschnitt 2.3 gezeigt haben, redigieren Autoren jedoch bereits, *bevor* der aktuelle Satz beendet ist. Wir finden hier ähnliche Kritikpunkte, wie sie Korrektoren für *Editor's Assistant* geäussert hatten: Die Benutzer werden zu stark eingeschränkt.

Die Autoren schliessen 1986 ihren Bericht optimistisch:

> The kind of text editing facilities provided by Author Environments will make part of any office workstation that truly deserves the epithet of Fifth Generation. Since the required technology departs considerably from current text editing software, there is a second reason for giving Author Environments the Fifth Generation epithet: namely as successor to the four generations of handwriting, printing, typewriting and word processing. [Kempen et al. 1986, S. 372]

Spätere Berichte erwähnen jedoch nur Prüf- und Korrekturprogramme, die erfolgreich implementiert wurden. Diese Werkzeuge wurden so implementiert, dass sie interaktiv benutzt und auch als explizite Nachbearbeitungsfunktionen verwendet werden konnten [siehe Kempen und Vosse 1992]. Die vorgeschlagenen Funktionen zur Pluralisierung von Substantiven und der automatischen Korrektur damit verbundener Seiteneffekte (verlorene Kongruenz mit Verben und Anaphern) sowie die Verschmelzung oder Trennung von Sätzen [Kempen et al. 1986, S. 371] wurden nicht implementiert.

19. Hier handelt es sich also um eine der wenigen Entwicklungen, die sich nicht mit Englisch beschäftigen.

3.3 Automatische Prüf- und Korrekturprogramme

Die oben beschriebenen Projekte und Ideen stammen alle aus den 1980er und 1990er Jahren. Einige der vorgestellten Projekte (etwa *Writer's Workbench*) sind Vorläufer der heute in Textbearbeitungsprogramme integrierten Prüf- und Korrekturprogramme. Wir gehen hier näher auf solche integrierten Programme ein.

Sind die Unterstützungen für das Schreiben selbst oder die verfügbaren Funktionen zum Redigieren eines Textes nicht optimal, schleichen sich Fehler ein. Semantische und logische Fehler sind nur bei konzentriertem (Korrektur)Lesen zu entdecken, syntaktische oder orthographische Fehler fallen hingegen schneller auf. Syntaktische und morphologische Gesetzmässigkeiten einer Sprache lassen sich gut in Regeln fassen. Daher liegt es nahe, entsprechende Überprüfungen vom Computer ausführen zu lassen und computerlinguistische Ressourcen dafür zu verwenden.

Es existieren verschiedene Prüf- und Korrekturprogramme, die einen Text hinsichtlich Orthographie, Grammatik und Stil kontrollieren und gegebenenfalls korrigieren oder Korrekturvarianten vorschlagen. Orthographieprüfprogramme verwenden orthographische und morphologische Regeln einer Sprache – führen also Überprüfungen auf Wortebene durch –, Grammatikprüfprogramme verwenden die syntaktischen Regeln einer Sprache und führen Überprüfungen auf Satzebene durch. Sie berücksichtigen hauptsächlich Kongruenzfehler. Stilprüf- und -korrekturporgramme beziehen sich nur zum Teil auf analysierbare syntaktische Strukturen. Unter «Stil» werden Aussagen über die Satzlänge (in Wortformen), die Wortlänge (in Buchstaben), die Frequenz bestimmter Wortarten, die Länge von Substantivgruppen, das numerische Verhältnis von Aktiv- und Passivkonstruktionen, die Komplexität der Satzstrukturen (also das numerische Verhältnis von einfachen Sätzen zu Satzverbindungen oder Satzgefügen), die Anzahl der Präpositionalphrasen, die Wortwahl, die Anzahl möglicher Referenten für ein Pronomen, die Darstellung von Zahlen als Wörter etc. verstanden, siehe die Darstellungen von Dale und Douglas [1996], Johnson und Guilfoyle [1989], Montero und Duque [2003], Smithson [1986], Vernon [2000] sowie Williams und Holt [1989].

Wir verwenden den Begriff *Prüf- und Korrekturprogramm* als Oberbegriff für alle Typen und werden nur bei Bedarf differenzieren, da «[…] la frontière entre correcteur orthographique et correcteur grammatical tend petit à petit à s'estomper dans les versions les plus récentes de ces outils […]» [Fontenelle 2005, S. 119].

Zunächst beschäftigen wir uns in Abschnitt 3.3.1 mit der Funktionsweise von Prüf- und Korrekturprogrammen. Nachdem wir in Abschnitt 3.3.2 die Korrektheit und Zuverlässigkeit solcher Programme erörtern, widmen wir uns in Abschnitt 3.3.3 dem Umgang von Autoren mit Prüf- und Korrekturprogrammen.

3.3.1 Funktionsweise

Ross [1991] beschreibt am Beispiel eines Stilprüfprogramms, wie Prüf- und Korrekturprogramme arbeiten *sollten*:

> A proper style-checking program should begin a text analysis by highlighting all the phrases or words in its repertoire […] The menu could also include (successivly) a possible revision of each error and an explanation. With this approach, the passive handbook would be replaced with an active look-up procedure sensitive to context, written in nontechnical language, and backed up with examples and tutorials. [Ross 1991, S. 98f]

Diese Beschreibung ist auch fast 20 Jahre später nur ein Wunschbild. Vielmehr herrscht der Zustand, den Vernon [2000, S. 331f] beschreibt: Prüf- und Korrekturprogramme seien allgegenwärtig und vor allem wenig zuverlässig. Elaborierte Versuche aus den 1980er Jahren hätten sich nicht durchgesetzt und seien in Vergessenheit geraten. Kommerzielle Produkte, und sogar in Textbearbeitungsprogramme eingebaute, seien seit Ende der 1980er, Anfang der 1990er Jahre sehr viel billiger. Ähnliches können wir auch für Textbearbeitungsprogramme generell feststellen, siehe Abschnitt 3.1.1.

Die heute verfügbaren Prüf- und Korrekturprogramme in Textbearbeitungsprogrammen können in der Regel in zwei Modi ausgeführt werden: durch Aufruf der entsprechenden Funktion – dann wird der aktuelle Text komplett überprüft – oder *während* des Schreibens (z. B. in *MS Word* durch Aktivieren der Option `check-as-you-type`) ohne expliziten Funktionsaufruf. Das Prüf- und Korrekturprogramm markiert «problematische» Textelemente – bekannt sind die roten (Orthographie) und grünen (Grammatik und Stil) Unterkringelungen – und gibt dem Benutzer Informationen über die Art des Problems und Hinweise zu dessen Behebung. Dabei werden die Regeln angewandt, die der Sprache entsprechen, die für das Prüf- und Korrekturprogramm aktiviert wurde.

Hier liegt das erste Problem: Das Prüf- und Korrekturprogramm kann nicht automatisch die Sprache des zu überprüfenden Textes erkennen. Ist «deutsch» aktiviert – dies kann sowohl für das Textbearbeitungsprogramm selbst geschehen als auch für die Überprüfung allein –, wird jedoch englisch geschrieben, werden alle nicht-deutschen Wörter rot und/oder grün markiert. In Bezug auf Stil können Einstellungen zu Genres und damit verbundenen Werten zu «erlaubten» oder «erwünschten» Satzlängen, Satzkonstruktionen etc. vorgenommen werden. Hier liegt das nächste Problem: Diese Einstellungen sind schwer zu finden und können nicht beliebig vorgenommen werden, die Werte für die Satzlänge können beispielsweise nur aus einer Liste ausgewählt werden.

3.3.2 Korrektheit

Verschiedene Studien zeigen, dass Prüf- und Korrekturprogramme nicht korrekt und vor allem nicht zuverlässig arbeiten: Vernon hat 2000 das integrierte Prüf- und Korrekturprogramm von *Word Perfect* mit 36 typischen Fehlern des Englischen – in zumeist sehr kurzen Sätzen – getestet. Lediglich ein Drittel dieser Fehler wurde erkannt, zu einigen wurden falsche Lösungen zur Behebung

angeboten, für einige konnte keine Lösung angeboten werden. [Vernon 2000, S. 340ff] Er kommt zur Erkenntnis:

> Most significantly, the checkers are restricted to the sentence level. They cannot identify inconsistent verb tenses or vague pronoun references across terminal punctuation. Checking such items across sentences may be impossible, because it requires contextual knowledge beyond the checker's purely structural knowledge. Unfortunately, a fair portion of novice writer's errors are exactly these cross-sentence culprits. [Vernon 2000, S. 340]

Und weiter heisst es: «Checkers frequently flag a troubled sentence but incorrectly identify the issue. Something is wrong, they just can't determine what.» [Vernon 2000, S. 344] Insgesamt ist dies also eine sehr unbefriedigende Situation.

Einige Jahre später stellt Hirst fest, dass die Orthographieprüfung in MS *Office 2007* für Englisch «[…] fails to find most errors, but when it does flag a possible error, it is almost always correct.» [Hirst 2008, S. 1] Hirst untersucht einen Teil des «Wall Street Journal», in das über Regeln automatisch 1'306 Schreibfehler (als *real-world errors*) eingefügt wurden. Entsprechend der Werbung von *Microsoft* sollten solche gefunden und korrigiert werden, da MS *Office 2007* erstmals über einen *contextual spelling checker* verfüge. Es wurde jedoch weniger als ein Viertel dieser Fehler gefunden. Der Korrekturvorschlag für diese war dann jedoch nahezu immer korrekt und es wurden lediglich 11 Wörter fälschlicherweise als Fehler markiert. [vgl. Hirst 2008, S. 4] Die Trigramm-Methode, wie von Wilcox-O'Hearn et al. [2008] beschrieben, die im Gegensatz zu den Prüfprogrammen in MS *Office* keine linguistischen Ressourcen benutzt, schneidet deutlich besser ab und findet 40 % aller Fehler [Hirst 2008, S. 7].

Die Situation ist für Sprachen mit einer reicheren Morphologie und Syntax als Englisch komplexer: Die Entwicklung der Prüf- und Korrekturprogramme stagniert und nicht nur Anfänger stehen vor dem Problem, nicht zu wissen, wie zuverlässig die Diagnosen sind. Von 70 der von uns in Anhang A gesammelten Fehler wurden vom *Grammar and Style Checker* von MS *Word* (MSGC) lediglich acht als Fehler erkannt, für nur sechs dieser acht wurde der richtige Korrekturvorschlag geliefert, für einen wurde ein falscher Vorschlag gemacht, für einen wurde nichts vorgeschlagen [siehe Gubelmann 2010].

Diese Zahlen bestätigen den schlechten Eindruck, den Kramer [2004] in seiner Magisterarbeit für die Überprüfung und Korrektur von Orthographiefehlern in einem grösseren deutschen Zeitungskorpus gewinnt: Die getesteten Prüf- und Korrekturprogramme *DITECT – Rechtschreibprüfung*[20] und MSGC markieren etwa zehnmal soviele Fundstellen, wie tatsächlich Fehler vorhanden sind [Kramer 2004, S. 47]. Kramer ermittelt eine Ausbeute (*recall*) von etwa 30 % und eine Präzision (*precision*) von etwa 4 % [Kramer 2004, S. 49].

Prüfprogramme sind auf typische Fehler von Muttersprachlern ausgerichtet. Die Erkennung und Korrektur von Orthographiefehlern, die beispielsweise englische Muttersprachler im Deutschen machen, werden zu 62 % «korrigiert» –

20. http://www.ub-dieck.com/ditectd.htm (zuletzt besucht am 8.12.2010, 19:34), vertrieben von der Unternehmensberatung Dieckmann, vor allem in Verlagen im Einsatz.

d. h., die korrekte Form ist unter den vorgeschlagenen Verbesserungen für einen Fehler –, wie Rimrott und Heift [2008] anhand von Texten von Sprachlernern zeigen. Wobei Wörter mit nur einem Fehler in 90 % der Fälle richtig korrigiert wurden, Wörter mit mehr als einem Fehler jedoch nur in 5 % der Fälle. 6 % aller Fehler wurden nicht erkannt [Rimrott und Heift 2008, S. 82].

Eine Ursache der schlechten Ergebnisse liegt darin, dass es leichter ist, einen Fehler festzustellen, als diesen Fehler zu korrigieren [Dale 1989, S. 9f]. Dies trifft sowohl für die automatische Überprüfung als auch auf die Überprüfung durch Menschen zu [Largy et al. 2004]. Oft sind mehrere Varianten der Korrektur möglich, die jedoch eventuell die Aussage der betroffenen Wortgruppe oder des Satzes verändern. Eine vollautomatische Korrektur ist also gar nicht möglich.

Dale [1989] konstatiert, dass Prüf- und Korrekturprogramme im Bereich Orthographie gute Ergebnisse lieferten, der Benutzer eines solchen Systems jedoch sehr viel Wissen über die entsprechende Sprache brauche. Entscheidend für die Akzeptanz bei den Benutzern sei die Bedienoberfläche [Dale 1989, S. 8]. Fast zehn Jahre später schreibt er: «[…] the techniques used here are now fairly stable, although without major advances (for example, taking explicit account of syntax and even semantics) we cannot expect much beyond current performance.» [Dale 1997, S. 236] Dies bestätigen die Experimente von Hirst [2008], wie oben beschrieben.

Die Überprüfung von Grammatik ist erheblich anspruchsvoller. 1997 sind die aktuellen Prüfprogramme noch immer stark beeinflusst von *Writer's Workbench*[21], dem schon damals 20 Jahre alten Prüf- und Korrekturprogramm. Neuere Entwicklungen nutzten lediglich bessere Hardwarevoraussetzungen aus [Dale 1997, S. 236] und enthielten weit weniger computerlinguistisches Know-How als die Werbung jeweils verspreche [Dale und Douglas 1996, S. 123].

Die Überprüfung oder Korrektur von «Stil» stützt sich auf Empfehlungen zu gutem Schreiben, die sich jedoch oft nicht in Regeln fassen lassen, da sie nicht eindeutig interpretiert werden können, wie etwa die Forderung, «konkret statt abstrakt» zu schreiben, siehe [Dale 1989, S. 12]. Ähnlich urteilen Sanford und Moxey [1989]: Es sei nicht möglich zu definieren, wie Stil-Regeln formuliert werden müssten, wie es auch nicht möglich sei, exakt zu definieren, wie Rückmeldungen über allfällige Verletzungen dieser Regeln erfolgen sollten und welche Hinweise zur Behebung zu geben seien. Ebensowenig sei es möglich, dies zu programmieren [Sanford und Moxey 1989, S. 44ff]. Die Gründe lägen in nicht verfügbaren psycholinguistischen Erkenntnissen.

Sanford und Moxey schlagen als Ersatzlösung einen interessanten Medienbruch vor: Das Geschriebene solle (automatisch) vorgelesen werden, so würden Stilverletzungen leichter erkannt und könnten dann durch den Autor verbessert werden [Sanford und Moxey 1989, S. 46f]. Dass dies prinzipiell möglich ist und zu akzeptablen Resultaten führt, bestätigen Leijten et al. [2010b] und Pilotti et al. [2004, 2006] für Korrekturen von Orthographie- und Grammatikfehlern.

Wenn also Prüf- und Korrekturprogramme nicht zuverlässig arbeiten und die Resultate als mangelhaft einzuschätzen sind, wie gehen Autoren mit diesen Werkzeugen um? Welchen Einfluss haben sie auf den Schreibprozess?

21. Siehe Abschnitt 3.2.2.

3.3.3 Umgang mit Prüf- und Korrekturprogrammen

Während sich Vernon hauptsächlich mit *Word Perfect* beschäftigte, geht es bei McGee und Ericsson [2002] wie bei Kramer um den *Grammar and Style Checker* von *MS Word*. Sie stellen fest, dass – vor allem mit Benutzung der Option `check-as-you-type` – Fragen von Grammatik, Orthographie und «Stil» den Hauptfokus beim Schreiben erhielten und damit der Schreibprozess beeinflusst werde. Kaum jemand kenne und nutze die Möglichkeit, verschiedene Vorgaben für «Stil» zu verwenden oder anzupassen. [McGee und Ericsson 2002, S. 454 und S. 462] Auch wenn MSGC computerlinguistisch auf einem sehr hohen Niveau sei (vgl. auch die Ausführungen von Heidorn [2000] und Fontenelle [2005]), konstatieren sie: «Paradoxically, however, its increased sophistication may make it more dangerous than ever before.» [McGee und Ericsson 2002, S. 456] Die *optische* «Professionalität» von MSGC impliziert *inhaltliche* Professionalität. McGee und Ericsson [2002] ermitteln drei Arten von Reaktionen darauf:

- Autoren *ignorieren* die roten und grünen Markierungen – was jedoch nicht identisch ist mit «nicht beeinflusst sein von etwas».
- Autoren versuchen *einige* Markierungen zu ignorieren – z. B. diejenigen in Bezug auf Satzlänge.
- Autoren passen sich an die Vorgaben des verwendeten Stils und der Grammatik- und Rechtschreibregeln an, indem sie versuchen, Markierungen zu *vermeiden*.

Die Meldungen des Prüf- und Korrekturprogramms werden als absolut und unumstösslich empfunden [Heilker 1992, S. 65]. Klar ist, dass MSGC damit das Schreiben und auch das Geschriebene beeinflusst [McGee und Ericsson 2002, S. 462], also sowohl Auswirkungen auf den Prozess als auch auf das Produkt hat. Als besonders gefährlich erscheint dabei die Absolutheit der Markierungen. Im Gegensatz zu menschlichen Korrektoren lassen die angewendeten Regeln keinen Spielraum zu. Das «Spielen» mit Sprache, um die beste Variante zu finden, wird behindert.

Neben dem Ignorieren von Markierungen oder Meldungen – was kognitive Kapazität kostet, die für den eigentlichen Schreibprozess dann nicht mehr zur Verfügung steht – wählen vor allem professionelle Autoren die Option, Prüf- und Korrekturprogramme gar nicht zu verwenden: «[…] professional writers did not regard style or grammar-checkers highly» [Dorner 1992, S. 7]. Die Voreinstellung in Textbearbeitungsprogrammen ist in der Regel die Verwendung von Prüf- und Korrekturprogrammen mit der Option `check-as-you-type`, ein Autor muss die Prüf- und Korrekturprogramme also bewusst abschalten.

Duffy und Robinson [1996] stellen in ihrer Studie mit 27 erfahrenen Korrektoren und Lektoren fest, dass die wenigsten (lediglich zwei) Prüf- und Korrekturprogramme verwenden und diese sehr negativ beurteilen:

> One editor commented that the tools force an extra process in the editorial cycle at the cost of time and money. The primary reaction from the users was that the tools have “some good points” but in general the “time it takes to go through those [comments] that are

> invalid and irrational makes it not worth the time of effort". [Duffy und Robinson 1996, S. 15]

In der Liste der Wünsche für Schreibunterstützung in Textbearbeitungsprogrammen rangieren Prüf- und Korrekturprogramme sehr weit hinten [Duffy und Robinson 1996, S. 18]. Die wichtigste Anforderung an Prüf- und Korrekturprogramme ist das Finden von Fehlern – und zwar von tatsächlichen Fehlern – *ohne* eine automatische Korrektur:

> […] we suspect that the editors will be able to generate a correction faster than they can evaluate computer proposed modifications – and that the proposed modifications would probably have to be modified in many instances. So even suggesting alternatives may be counterproductive. [Duffy und Robinson 1996, S. 21]

McGee und Ericsson gehen sogar noch weiter und bezichtigen MSGC der Verursachung von Schreibblockaden, da vier der von Rose hierfür definierten Ursachen Beschreibungen des Verhaltens von MSGC seien:

> […] editing too early in the composing process
>
> […] the rules by which they guide their composing process are rigid, inappropriatly invoked, or incorrect.
>
> […] they invoke conflicting rules, assumptions, plans, and strategies and
>
> […] they evaluate their writing with inappropriate criteria or criteria that are inadequatly understood. [Rose 1983, S. 4]

Vernon wie auch McGee und Ericsson lehnen die Benutzung von Prüf- und Korrekturprogrammen jedoch nicht vollständig ab. Sie propagieren eine Verwendung in der Lehre, die die Meldungen inklusive Erklärungen und Korrekturvorschläge als Ausgangspunkt benutzt, um über einen Text nachzudenken und sich mit Fragen von Stil und Grammatik auseinanderzusetzen: «[…] the teacher can use the grammar checker to introduce students to the very problematic nature of grammatical authority and normative language.» [Vernon 2000, S. 333] Das Prüfprogramm übernimmt also die Markierung von syntaktischen Elementen, entsprechend dem Vorschlag von Eyman und Reilly [2006], die Möglichkeiten von Textbearbeitungsprogrammen zur Hervorhebung einzelner Elemente zu verwenden, um den Redigierprozess explizit zu unterstützen (siehe die Ausführungen im Abschnitt 2.3.2). Ähnlich merkt Blatt an, «[…] dass die in Textverarbeitungsprogrammen integrierten Schreibhilfen nicht automatisch verbessern, sondern dass sie didaktisch sinnvoll genutzt werden müssen.» [Blatt 2004, S. 37]

Sicher ist dies nicht die Verwendung, für die MSGC und vergleichbare Prüf- und Korrekturprogramme in Textbearbeitungsprogrammen entwickelt wurden. Das Ganze ist ein wenig vergleichbar mit den Hoffnungen, die in die ersten elektronischen Lernunterstützungen gesetzt wurden: Die Maschine solle den «Rotstift» des Lehrers übernehmen, damit der Lehrer sich mehr auf eine beratende Tätigkeit konzentrieren könne und nicht nur derjenige sei, der Fehler

anstreicht. Dies führte unter anderem zur Entwicklung von *TICCIT* (time-shared interactive, computer-controlled, information television) [Whithaus 2004]. Jedoch:

> In TICCIT's case, market considerations combined with behavioral science methodologies led to the creation of a technological system that focused on writing instruction as formal grammar correction and on communication content developed by experts not by the student writers or their teachers. [Whithaus 2004, S. 159]

Die Empfehlungen von Vernon decken sich mit denen, die Mortenson [1987] und Day [1988] bereits Ende der 1980er Jahre für die Verwendung von Programmen zur Nachbearbeitung von Texten geben (vgl. Abschnitt 3.2): «What you do, with the report generated […] is up to you. These programs cannot make poetry out of sludge. The content has to come from the writer.» [Mortenson 1987, S. 69] «The writer is always in charge and must decide whether or not to make changes. When to make changes, and what changes to make, must be determined by criteria not specified by the [checker – C.M.]» [Day 1988, S. 73] Der Computer ist sehr gut geeignet, bestimmte Informationen über oder aus einem Text zusammenzutragen – besonders lange Wörter, Interpunktion, koordinierende und subordinierende Wörter, Frequenzen für Wörter mit einer bestimmten Länge, Anzahl der Silben je Wort etc. [Mortenson 1987, S. 70ff] – die Entscheidungen, was entsprechend diesen Informationen geändert werden sollte, muss und kann jedoch nur der Benutzer treffen. Die Programme sind nicht dazu gedacht, einen Text zu verändern, dies bleibt Aufgabe des Autors.

Dieser Spielraum ist für Benutzer von aktuellen Prüf- und Korrekturprogrammen in Textbearbeitungsprogrammen stark eingeschränkt. Wird auf die Option `check-as-you-type` verzichtet und die Überprüfung *nach* einer Schreibphase explizit ausgelöst, wird der Autor in eine sehr passive Position gedrängt. Das Prüf- und Korrekturprogramm untersucht den gesamten Text linear, meldet jede einzelne Fundstelle inklusive Erklärung des «Problems» und präsentiert Korrekturvorschläge. Es gibt keine Möglichkeit, einen Überblick über *alle* Arten von «Problemen» zu erhalten oder sich anzeigen zu lassen, wo im Text ein gleiches oder ähnliches «Problem» wie das gerade markierte auftritt.

Neben der schlechten Quote bezüglich gefundener tatsächlicher Fehler und der Korrektheit der entsprechenden Erklärungen und Korrekturvorschläge werden die Unflexibilität des Prüf- und Korrekturprogramms und fehlende Einflussmöglichkeiten als störend empfunden. Der Zeitfaktor für Analysen von automatischen Prüf- und Korrekturprogrammen spielte zwar zu Beginn der 1990er Jahre noch eine Rolle – Oakman vermutet, dass das Program *Epistle* von *IBM* aus den 1980er Jahren kommerziell nie erfolgreich war, da es zu rechenintensiv gewesen sei [Oakman 1994, S. 234] –, fällt heute jedoch nicht mehr ins Gewicht, der Benutzer muss nicht mehr mehrere Minuten auf die Ergebnisse einer Analyse warten. Trotzdem bleibt die Frage, ob sich der Einsatz der automatischen Prüf- und Korrekturprogramme lohnt und die Zeit für das Lesen der Analyseergebnisse aufgewandt werden soll, wenn die Ergebnisse so schlecht sind [vgl. Williams 1990b, S. 16f].

Johnson und Guilfoyle urteilen 1989 in ihrer Vorhersage über die Entwicklung von Prüfprogrammen in der Textbearbeitung in den nächsten Jahren eher negativ:

> They make crude checks on such matters as sentence length and the use of the passive, but they do not parse the texts they process. True – that is, parsing – text editing systems have not yet appeared in product form, and are not expected to do so for several more years. [Johnson und Guilfoyle 1989, S. 5]

Etwa 15 Jahre später hat sich diese Situation eigentlich nicht geändert, auch wenn diese nun eher optimistisch beurteilt wird:«Nevertheless, thorough computer-based style analyzers are still a matter of NLP research and there is much room for improvement.» [Montero und Duque 2003, S. 544]

Es ist jedoch fraglich, ob Prüf- und Korrekturprogramme innerhalb von Textbearbeitungsprogrammen in der heutigen Form für den Schreib- und Redigierprozess förderlich sind. Wir sprechen hier von deutschen Texten, die von erfahrenen Autoren verfasst werden, wobei sich «erfahren» auch auf die Sprache bezieht. Unzweifelhaft sind Prüf- und Korrekturprogramme positiver zu bewerten, werden sie von Autoren verwendet, die sich im Gebrauch der Sprache des zu verfassenden Textes nicht ganz sicher fühlen. In diese Richtung gehen die aktuellen Entwicklungen von *Microsoft* für Englisch, die ganz klar Nichtmuttersprachler als Zielpublikum haben, siehe die Beiträge von Gamon [2010] und Rozovskaya und Roth [2010] oder `http://www.eslassistant.com/` (zuletzt besucht am 8.12.2010, 19:34). Dazu passt die Managemententscheidung von *Microsoft*, MSGC in absehbarer Zeit nicht weiterzuentwickeln. *Microsoft* konzentriert sich stattdessen auf die Entwicklung von Rechtschreibprüfung für weitere Sprachen.[22]

In jedem Fall können Prüf- und Korrekturprogramme lediglich Fehler im bereits geschriebenen Text erkennen. Solche Programme unterstützen nicht den eher kreativen Aspekt des Redigierens, der sich auf das Ausprobieren verschiedener Formulierungen bezieht und Veränderungen am Text erfordert, weil sich die kommunikative Situation oder Idee ändert oder der Autor seine kommunikativen Absichten in eine andere Richtung lenkt. Die Fehler, die von Prüf- und Korrekturprogrammen gefunden werden (sollen), sind zu einem grossen Teil *slips*, deren Entstehung durch angemessene Funktionen vermieden werden könnte.

3.4 Weitere Schreibhilfen

Im Folgenden werfen wir einen kurzen Blick auf die Unterstützung, die nicht direkt mit dem Redigieren in Zusammenhang steht. Wir erwähnen hier beispielhafte aktuelle Projekte, um die Breite der Forschung im Bereich Computer und Schreiben aufzuzeigen.

3.4.1 *ANESTTE* – Unterstützung für das Schreiben wissenschaftlicher Texte

ANESTTE (*ANalyzEr of Style for Technical Texts in English*) ist ein Werkzeug zur Nachbearbeitung von Text von Autoren, die nicht englischer Muttersprache sind und wissenschaftliche Beiträge in Englisch verfassen. Die Prüfung eines

22. Persönliche Information von Takako Aikawa, Program Manager in der Natural Language Processing Unit von *Microsoft*, 1. Juni 2008.

solchen Textes hinsichtlich Orthographie und Grammatik ist explizit an andere Werkzeuge ausgelagert. Hier handelt es sich um eine reine Stilprüfung. [Montero und Duque 2003, S. 544f] Bearbeitet werden also Texte eines bestimmten Genres, die sich durch einen spezifischen Jargon auszeichnen. Sie werden hinsichtlich der textlinguistischen Dimensionen Klarheit (*clarity*), Prägnanz (*conciseness*), Variation (*variety*) und Überzeugungskraft (*conviction*) untersucht. Da automatische semantische Analyse zu wenig genau und zuverlässig sei, wird für die Ermittlung von Kohärenz und Kohäsion lediglich die syntaktische Struktur der Sätze verwendet. [Montero und Duque 2003, S. 546]

Das System arbeitet regelbasiert; die Regeln wurden manuell aus verfügbaren Stilvorgaben für englische wissenschaftliche Texte erstellt. Die Regeln werden nacheinander auf den Text angewandt [Montero und Duque 2003, S. 547]. Die Funktionsweise ist also sehr ähnlich der des *Editor's Assistant* (siehe Abschnitt 3.2.5), auch wenn von den Autoren kein Bezug dazu hergestellt wird. Der Benutzer wird anschliessend mit einem animierten Agenten durch den erstellten Report geführt.

Leider berichten die Entwickler nicht über die Nützlichkeit oder Korrektheit ihrer Analysen. Sie erwähnen lediglich, dass sie die Korrektheit der linguistischen Analysen überprüft hätten und mehr an Präzision als an der Maximierung der Fundstellen (*recall*) interessiert seien [Montero und Duque 2003, S. 549] – eine Aussage, der unbedingt zuzustimmen ist. Jedoch werden keine Ergebnisse solcher Experimente dargestellt. Das System wurde an bereits publizierten Texten von Nichtmuttersprachlern getestet. Hierfür wird berichtet, dass die Qualität dieser Texte – entsprechend den Analysen von *ANESTTE* – der Qualität der Texte von Muttersprachlern sehr nahe komme. [Montero und Duque 2003, S. 550] Dies sagt leider nichts über die Nützlichkeit von *ANESTTE* für einen Autor aus, der einen Text erst noch bearbeiten möchte oder muss, bevor er ihn zur Publikation einreicht.

3.4.2 *Flexpansion* – Abkürzungen erweitern

Personen mit physischen Handicaps oder auch Personen, die möglichst wenig physische Schreibarbeit leisten wollen, benötigen Hilfe bei der Eingabe von Text. Eine Möglichkeit ist die Verwendung von Spracherkennungssoftware, um Diktate zu verschriftlichen, wie Leijten et al. [2010a] darstellen.

Eine andere Variante ist, möglichst wenige Buchstaben über die Tastatur einzugeben und daraus vollständige Sätze zu erhalten, siehe [Willis et al. 2002, S. 251]. Dabei sind verschiedene Strategien vorstellbar: (a) das Weglassen von Vokalen, (b) die Verwendung von Wortstämmen, (c) das Ausnutzen phonetischer Besonderheiten einer Sprache (gleichlautende Buchstaben(kombinationen) werden nur durch einen Repräsentanten dargestellt) oder (d) personalisierte Varianten, die entsprechend festgehalten und trainiert werden müssen [Willis et al. 2002, S. 257].

Es existieren bereits Werkzeuge, die (a) «Wörter vorhersagen», also nach wenigen eingegebenen Buchstaben Kandidaten für das gemeinte vollständige Wort zur Auswahl anbieten und dabei das bereits Geschriebene einbeziehen oder (b) Abkürzungen erweitern. Eine sehr einfache Variante *ohne* Einbezug des Kontexts ist *T9* für das Schreiben von SMS auf dem Mobiltelefon.

Der Ansatz der adaptiven Expansion von ad hoc erstellten Abkürzungen für Englisch, den Willis [2008] in seiner Dissertation ausführlich darstellt, wird mittlerweile als *Flexpansion* für verschiednee Geräte von seiner Firma angeboten, siehe http://www.flexpansion.com/ (zuletzt besucht am 8.12.2010, 19:34).

3.4.3 Die richtigen Worte finden

Soll ein Text oder Textabschnitt umformuliert werden, kann der Autor beim Finden von Formulierungen unterstützt werden, wie Max und Zock [2008] zeigen. Sie fokussieren auf das Ersetzen von Wortgruppen und schlagen ein Lexikon von Wortgruppenentsprechungen vor, das eine zweite Sprache als *pivot* verwendet, um Paraphrasierungen zu generieren. Das System wurde für Französisch mit Englisch als *pivot* implementiert und evaluiert [Max und Zock 2008, S. 81f].

Lapalme und Macklovitch [2008] schlagen vor, automatische Ergänzungen für das Schreiben in der Fremdsprache zu erzeugen, haben ein solches System jedoch nicht implementiert. Autoren, die in einer Fremdsprache schreiben, wären toleranter gegenüber automatisch erzeugten Fortsetzungsvorschlägen. Ebenfalls Unterstützung für das Schreiben in einer Fremdsprache (Englisch) bietet *Netspeak*[23] [Stein et al. 2010]. Hier werden typische Kollokationen etwa für die Verwendung von Präpositionen vorgeschlagen.

3.4.4 *Writer's Aid*, *Remembrance Agent*, *À Propos* – Referenzieren und zitieren

Writer's Aid unterstützt den Autor dabei, während des Verfassens wissenschaftlicher Texte Referenzen zu suchen, die das gerade Geschriebene belegen können. Der Computer sei laut Babaian et al. [2002] besser geeignet, Referenzen zu suchen, da er systematischer vorgehe, Zeit keine Rolle spiele – der Suchprozess interferiert für den Autor mit dem eigentlichen Schreiben, für den Computer jedoch nicht – und bei Misserfolgen in regelmässigen Zeitabständen die Suche wiederholt werden könne. Entwickelt wurde das Werkzeug für *Emacs*, Zielpublikum sind Autoren, die mit LaTeX und BibTeX arbeiten.

Grundlage für die Suche sind Stichwörter, die der Autor erzeugt. Anschliessend wird in vorhandenen BibTeX -Dateien auf dem Rechner des Autors, online in ACM-Daten und in Daten des *NEC Research Institute* nach entsprechenden Quellen gesucht. Die Ergebnisse werden im *Emacs* in einem separaten Fenster fortlaufend angezeigt – sobald ein Suchprozess erfolgreich war, wird die Liste der Fundstellen ergänzt. Diese können editiert und als Referenz in den aktuellen Text und in die zugehörige BibTeX -Datei aufgenommen werden. Wenn möglich, wird dem Autor zusätzlich eine PDF-Version der gefundenen Literatur angeboten.

Ziel ist hier ganz klar, den Autor von notwendigen, aber zeitaufwändigen und wenig kreativen Aufgaben zu entlasten. Ähnliche Ziele hat auch *Remembrance Agent*. Hier werden, ausgehend von der aktuellen Tätigkeit des Benutzers, Verbindungen zu bereits vorhandenen Dokumenten gesucht. Dabei werden E-Mails des Benutzers und weitere indexierte Dokumente verwendet [Rhodes

23. http://www.netspeak.cc/ (zuletzt besucht am 8.12.2010, 19:34).

und Starner 1996]. *Remembrance Agent* ist ebenfalls für *Emacs* implementiert und für verschiedene *UNIX*-basierte Betriebssysteme frei verfügbar.[24]

Die Idee von *Remembrance Agent* entwickeln Puerta Melguizo et al. [2008a,b] weiter und stellen mit *À Propos* (auch als *IntelliGent* bezeichnet) eine proaktive Suchunterstützung vor, die möglichst nicht den Schreibprozess unterbrechen soll. Gefordert wird grösstmögliche Korrektheit für die Suchergebnisse und die Berücksichtigung des aktuellen Schreibprozesses sowohl hinsichtlich der benötigten Informationen als auch hinsichtlich deren Präsentation [Puerta Melguizo et al. 2008b, S. 23]. Während des Schreibens werden Phasen des Planens und des Redigierens anhand der Interaktion des Autors mit der Tastatur bzw. dem Textbearbeitungsprogramm automatisch bestimmt, nur innerhalb dieser Phasen werden Fundstellen in einem transparenten Fenster in *MS Word* angezeigt.

3.4.5 *Scrivener* – Organisation des Schreibprozesses

Mit *Scrivener*[25] werden Applikationen für verschiedene Phasen des Schreibprozesses vom Planen bis zum Redigieren in ein Produkt integriert. Hier werden alle unterstützenden Arbeiten um ein existierendes Textbearbeitungsprogramm (*TextEdit*) gruppiert, um dem Autor verschiedene Sichten auf das zu bearbeitende Material, seinen Planungsprozess, zu integrierende Referenzen etc. zu ermöglichen. Das eigentliche Eingeben und Bearbeiten des Textes wird nicht gesondert unterstützt, es stehen die normalen Funktionen von *TextEdit* für *MacOSX* zur Verfügung.

3.4.6 *Final Draft* – Genrespezifische Editoren

Unterschiedliche Genres stellen verschiedene Anforderungen an die Struktur, die Sprache und oft auch an das Layout eines Textes. Augenfällig ist dies für Drehbücher, die auch im 21. Jahrhundert aussehen, als ob sie mit der Schreibmaschine erstellt worden wären – hier ist sogar die verwendete Schriftart entscheidend. Drehbücher weisen zudem eine spezielle Strukturierung des Textes auf. Die beteiligten Figuren sind von vornherein festgelegt und müssen bei jeder Handlung oder Äusserung explizit erwähnt werden. *Final Draft*[26] unterstützt Drehbuchautoren bei der äusseren Gestaltung eines Drehbuchs, bei der Eingabe von Figurennamen, der Angabe von Regieanweisungen etc. Strukturelemente – einzelne Szenen oder Dialoge – können komfortabel hervorgehoben oder etwa im Gesamttext verschoben werden. *Final Draft* erlaubt es Autoren also, auf genrespezifische Strukturen zuzugreifen und auf diesen zu operieren.

3.5 Zusammenfassung

In diesem Kapitel haben wir Systeme zur Unterstützung des Schreibprozesses vorgestellt, die teilweise computerlinguistische Methoden und Ressourcen verwenden, um einen Aspekt des Redigierens – die Erkennung, Diagnose und Korrektur von Fehlern – zu erleichtern, oder die auf andere spezielle Phasen des

24. http://www.remem.org/ (zuletzt besucht am 8.12.2010, 19:34).
25. http://www.literatureandlatte.com/scrivener.html (zuletzt besucht am 8.12.2010, 19:34).
26. http://www.finaldraft.com/products/final-draft/ (zuletzt besucht am 8.12.2010, 19:34).

Schreibprozesses fokussieren. Wir finden jedoch keine Systeme, die es Autoren erlauben, auf sprachliche Einheiten ihrer Texte zuzugreifen und auf diesen zu operieren. Funktionen zur Textbearbeitung sind zeichenbasiert und operieren nicht auf linguistischen Einheiten. Daher müssen Autoren ihre Redigierziele, die in der Regel unter Verwendung linguistischer Begriffe formuliert werden können, auf komplexe und lange Sequenzen solcher zeichenbasierten Funktionen herunterbrechen. Dadurch werden kognitive Ressourcen beansprucht, die für die eigentliche Redigierphase nicht zur Verfügung stehen.

Zusammenfassend lässt sich sagen, dass seit Beginn des Einsatzes von Computerprogrammen als Schreibwerkzeug verschiedene Beobachtungen gleichbleiben:

- Textbearbeitungsprogramme bieten generell Funktionen, die das Schreiben unterstützen und die Umsetzung verschiedener Schreib- und Redigierstrategien ermöglichen. Autoren können unterschiedliche Schreibstrategien verfolgen.
- Textbearbeitungsprogramme und neben- oder nachgeordnete Werkzeuge bieten Funktionen, die den Autor beim Erstellen «guter» Texte unterstützen sollen durch Analysen hinsichtlich orthographischer oder syntaktischer Aspekte. Deren Ergebnissen ist jedoch mit Vorsicht zu begegnen, der Autor sollte ihnen nicht blind vertrauen.
- Funktionen in Textbearbeitungsprogrammen und zugehörigen Werkzeugen sollten so «zurückhaltend» wie möglich sein. Sie sollen sich nicht aufdrängen, aber bei Bedarf zur Verfügung stehen.
- Die meisten Programme und Funktionen werden für Englisch konzipiert und entwickelt.

Jedoch fokussieren alle hier vorgestellten Programme zur Prüfung und Korrektur – auch wenn sie heute interaktiv eingesetzt werden können – auf die Behandlung von *Fehlern* und unterstützen nicht den gesamten Redigierprozess, wie wir ihn in Abschnitt 2.3.1 vorgestellt haben. Wir unterscheiden zwischen *Oberflächenkorrekturen* d. h., dem Beheben von orthographischen, grammatikalischen oder «mechanischen» Fehlern, wie Wortwiederholungen oder Interpunktion, und dem tatsächlichen *Ändern von Text* – also dem Ändern von Struktur, Inhalt oder Intention eines Textes. Inhaltliche Textänderungen erfordern Operationen, die auf Wörtern, Phrasen oder Sätzen ausgeführt werden, und sind nicht Korrekturen von orthographischen oder grammatikalischen Fehlern.

4 Das Konzept des *linguistisch unterstützten Redigierens*

> *For present-day writing tools the phrase "word processor" is a misnomer. They do not process words, but rather they carry out simple operations on strings of characters.*
>
> —Mike Sharples & Thea van der Geest, *The New Writing Environment: Writers at Work in a World of Technology*, 1996

> *The key here is to add intelligence and sophistication to provide* language sensitivity, *enabling the software to see a text not just as a sequence of characters, but as words and sentences combined in particular structures for particular semantic and pragmatic effect.*
>
> —Robert Dale, *Computer Assistance in Text Creation and Editing*, 1997

Im Kapitel 2 haben wir den aktuellen Stand der Schreibforschung bezüglich des Redigierens am Computer dargestellt und in Kapitel 3 heute verfügbare Editierfunktionen in Textbearbeitungsprogrammen erörtert. Programme und Funktionen zur Unterstützung des Redigierens oder allgemein zum Nachbearbeiten eines Textes gehen von einem Redigierbegriff aus, der sich auf das Erkennen und Korrigieren von Fehlern beschränkt. Zudem sind die meisten dieser Programme für Englisch entwickelt worden, während wir uns mit dem Schreiben deutscher Texte beschäftigen.

In Abschnitt 1.4 haben wir festgestellt, dass sich typische Fehler in Texten als *slips* nach Norman [1981] erklären lassen. Solche *slips* entstehen, weil gegenwärtige Textbearbeitungsprogramme nur über Funktionen verfügen, die den Autor zwingen, sein Redigierziel durch die Ausführung einer langen, komplexen Folge von zeichenbasierten Editorfunktionen zu erreichen.

In diesem Kapitel stellen wir das Konzept des linguistisch unterstützten Redigierens vor, das erlaubt, auf linguistischen Einheiten zu operieren. Zunächst erörtern wir in Abschnitt 4.1 die Problemstellung und stellen dann in Abschnitt 4.2 unseren Lösungsansatz vor. In Abschnitt 4.4 gehen wir auf Prinzipien ein, die für die Implementierung entsprechender Funktionen befolgt werden müssen. Diese Funktionen können entsprechend den von uns in Abschnitt 4.3 vorgeschlagenen Klassifizierung spezifiziert werden.

4.1 Problemstellung

In existierenden Textbearbeitungsprogrammen finden wir Funktionen, die den Redigierprozess als *Erkennen*, *Diagnostizieren* und *Korrigieren* von *Fehlern* entsprechend grammatischen Regeln einer Sprache oder stilistischen Vorgaben behandeln. Dies gilt unabhängig davon, ob diese Funktionen interaktiv benutzbar sind oder in einem eigenständigen Programm auf einen Text angewandt werden. Die meisten dieser Funktionen zielen auf die Fehlerkorrektur in orthographischer oder syntaktischer Hinsicht ab. Jedoch schliesst das Redigieren Textmodifikationen ein, die nicht mit Fehlern zusammenhängen, etwa die Modifikation von bereits geschriebenem Text, um einer veränderten kommunikativen Absicht des Autors zu entsprechen, eine andere Idee zu verfolgen, sich auf bereits Geschriebenes zu beziehen etc., wie wir in Abschnitt 2.3.1 gezeigt haben.

Erfahrene Autoren haben ein kommunikatives Ziel, Inhalt und Zielpublikum des zu schreibenden Textes sind bestimmt, im Umgang mit der verwendeten Sprache fühlen sie sich sicher, entsprechend den drei Meta-Ebenen, die Horning [2006, S. 118f] identifiziert. Anders als Anfänger haben erfahrene Autoren also eine sehr klare Vorstellung über den zu schreibenden Text als Ganzen: «Planning by beginners is opportunistic and driven by local constraints, while expert planning is strategic: the writer's goals determine the generation and organization of content.» [Andriessen et al. 1996, S. 253] Während des Schreibens nehmen erfahrene Autoren jedoch Änderungen am ursprünglichen Plan vor.

Das Redigieren erfahrener Autoren bezieht sich nur zu einem geringen Teil auf die Korrektur von grammatikalischen oder orthographischen Fehlern (Tippfehler eingeschlossen). Hier geht es vielmehr um den kreativen Aspekt des Schreibens, um das explorative Finden der besten Formulierung, um das Kürzen von Texten, um das Ändern des Fokus eines Textes. Autoren experimentieren mit verschiedenen syntaktischen Strukturen oder der Wortwahl, siehe auch [Sharples 1999].

Erfahrene Autoren können die vorzunehmenden Änderungen in der Regel in einfachen Redigieranweisungen formulieren, sodass ihnen selbst oder einem Koautor klar ist, welche Änderungen am Text vorzunehmen sind. Beispiele sind die Änderung der Zeitform im ganzen Text, das Einfügen oder Entfernen

von Modalverben, die Verwendung eines anderen Begriffes statt des ursprünglichen, die Änderung des verwendeten Verbs, das Auftrennen von Satzgefügen oder das Verbinden mehrerer Sätze. Alle diese Änderungen sind morphologische und syntaktische Entsprechungen einer beabsichtigten semantischen oder pragmatischen Änderung. Solche Beschreibungen haben wir bereits in Abschnitt 2.3.1 in den Kommentaren zu den Änderungen gesehen, die Obama an seinem Redemanuskript vorgenommen hatte.

Heutige Textbearbeitungsprogramme, die lediglich zeichenbasierte Funktionen anbieten, zwingen den Autor, die gewünschte Operation in eine komplexe Folge von zeichenbasierten Funktionen zu übersetzen und diese in der richtigen Reihenfolge auszuführen. Die Länge dieser Sequenzen und ihre Komplexität verursacht *slips* und führt letztlich zu Fehlern wie in Anhang A. Die Ausführung dieser Sequenzen beansprucht kognitive Ressourcen, die für andere Prozesse nicht mehr zur Verfügung stehen – der Autor vergisst seine ursprüngliche Absicht oder die anschliessend geplante, der Fokus geht verloren, es schleichen sich Fehler ein.

Wenn bestimmte während des Schreibens auszuführende Prozesse automatisiert werden können – etwa korrekte Rechtschreibung oder Tastaturschreiben, wie von Johansson et al. [2010], Kellogg [2008] oder Torrance und Jeffery [1999] gezeigt –, kann sich der Autor stärker auf inhaltliche Aspekte konzentrieren, was den Schreibprozess erleichtert. Die Übersetzung von gewünschten Redigieroperationen in zeichenbasierte Textbearbeitungssequenzen kann durch Übung natürlich auch zu einem gewissen Grad «automatisiert» werden – es liegen jedoch unseres Wissens keine empirischen Erkenntnisse der Schreibforschung vor, wie sich dies konkret auswirkt.[1]

Die kognitiven Anforderungen und die Anfälligkeit für *slips* würden jedoch wesentlich stärker reduziert, wenn dem Autor Funktionen zur Verfügung stünden, die diesen Übersetzungprozess überflüssig machten, indem sie auf den gleichen Strukturen operierten, die er selbst zur Formulierung seiner Redigierabsicht verwendet. Solche Funktionen sollten die linguistischen Elemente einer Sprache verwenden, ähnlich wie sprachbasierte Funktionen in Texteditoren die Strukturen der entsprechenden Programmiersprache verwenden und geeignete Funktionen ermöglichen.

Wir werden im folgenden Abschnitt 4.1.1 sprachbasierte Funktionen in Texteditoren vorstellen. Anschliessend zeigen wir in Abschnitt 4.1.2, dass dieses Prinzip auf Funktionen in Textbearbeitungsprogrammen übertragbar ist, dies bislang jedoch nicht umgesetzt wurde. Wir definieren so die Forschungslücke, die wir mit dieser Arbeit füllen.

4.1.1 Sprachbasierte Funktionen in Texteditoren

Unterstützung für Programmierer in spezifischen Editoren hat bereits eine lange Geschichte. In den 1970er und frühen 1980er Jahren wurde versucht, sprachbewusste oder sprachbasierte (engl. *language-aware*, *language-based*) Funktionen in Texteditoren zu verwenden und den Programmierer zu unterstützen, Code zu schreiben, der zu jedem Zeitpunkt syntaktisch wohlgeformt ist. Solche Editoren werden als *syntax-directed* bezeichnet, etwa *EMILY* [Hansen 1971], *Cornell* [Tei-

1. Wir gehen in Abschnitt 5.3 noch genauer darauf ein.

telbaum und Reps 1981], *PEN* [Allen et al. 1981], *JANUS* [Chamberlin et al. 1981], *PARSE* [Chusho et al. 1983], *Mentor* [Gouge et al. 1983, Lang 1986], *PAN* [Ballance et al. 1992, Van De Vanter et al. 1992] oder *CodeProcessor* [Van De Vanter und Boshernitsan 2000]. In der Regel entspricht die Struktur der so erstellten Dokumente einer Baumstruktur. Diese Editoren setzten die Programmierprinzipien des *stepwise refinement* und *structured programming* [Dijkstra 1972, Wirth 1971] um.

Eine Motivation zur Entwicklung solcher Editoren ist die Erkenntnis «Programs are not text; they are hierarchical compositions of computational structures and should be edited, executed, and debugged in an environment that consistently acknowledges and reinforces this viewpoint.» [Teitelbaum und Reps 1981, S. 563]. Die Eingabe von Programmcode Zeichen für Zeichen wurde als unökonomisch betrachtet [Teitelbaum und Reps 1981, S. 569]. Editoren wie *EMILY* erlaubten dagegen die Konstruktion eines Programms durch Kombination verfügbarer Templates und Strukturelemente [Hansen 1971, S. 524]. Beim Anlegen eines neuen Artikels im Editor *sds* wurde der Benutzer durch das Anlegen der notwendigen folgenden Strukturelemente geführt und konnte dann solche Elemente inhaltlich editieren [Fraser 1981, S. 18f].

Eine zweite Motivation ist ganz klar das *Vermeiden von Fehlern* während der Eingabe oder des Editierens von Programmen, indem nur syntaktisch wohlgeformter Text erzeugt werden kann [Hansen 1971, Teitelbaum und Reps 1981] und Editierfunktionen direkt auf den Strukturen der Programmiersprache operieren [Chusho et al. 1983, Khwaja und Urban 1993, Lang 1986]. Hansen verwendet Formulierungen, die auf die Beschreibung der Ursachen von *slips* zutreffen: «The user will forget how to do what he wants, what his files contain, and even—if interrupted—what he wanted to do.» [Hansen 1971, S. 527]. Die kognitive Kapazität des Benutzers ist begrenzt, einfache Kommandos statt komplexer Sequenzen von Funktionen helfen, Fehler bzw. *slips* zu verhindern [Allen und Scerbo 1983]. Funktionen sollen auf den Elementen und Strukturen der entsprechenden Sprache operieren; Meyrowitz und van Dam [1982a, S. 352] nennen solche Funktionen *target-specific operations*. Khwaja und Urban [1993, S. 232] betonen, dass die Grösse der *editing unit* sprachabhängig sei.

Allerdings stellte sich heraus, dass die interne Repräsentation – also die Vorstellung, die ein Programmierer über die zu implementierende Aufgabe hat – nicht notwendigerweise der syntaktischen Repräsentation des fertigen Codes entspricht und Programmierer beim Schreiben des Codes sich darum nicht immer entlang dieser syntaktischen Repräsentation bewegen [Neal 1987]. Programmierer fühlten sich durch syntaxbasierte Editoren zu stark eingeschränkt, vgl. [Wood 1981, S. 4] [Van De Vanter 1995, S. 253] und [Neal 1987, S. 99f]. Die Manipulation von Code, dessen textuelle Oberfläche verschiedenen syntaktischen Strukturen entspricht, sei nicht ohne weiteres möglich [Teitelbaum und Reps 1981, S. 569f]. Einfache Editieroperationen und die Eingabe von neuem Code seien unnötig kompliziert [Lang 1986, S. 50]. Die syntaktische Analyse des aktuellen Codes koste jeweils viel Zeit und verlangsame die Interaktion des Programmierers mit dem Editor [Morris und Schwartz 1981, S. 29f]. Die Umformung einer syntaktischen Struktur in eine andere (etwa der Wechsel von einer `while`- in eine `if`- oder `do`-Schleife) sei über rein zeichenbasierte Bearbeitung schneller [vgl. Morris und Schwartz 1981, Teitelbaum und Reps 1981].

Programmierer wollten nicht durch das System zu einem bestimmten Verhalten gezwungen werden, sondern erwarteten, dass der Editor unterstützende Funktionen anbietet; deswegen sind heutige Editoren nicht syntaxbasiert:

> The compromise that has emerged in modern programming editors is to maintain a textual representation, but to add syntactic and semantic assistance "on the side". This assistance is provided on a best-effort basis through such mechanisms as coloring, completion, and pop-ups. [Edwards 2005, S. 514]

Die Bezeichnungen *language-dependent*, *language-based*, *language-sensitive* oder *language-aware* beziehen sich heute also mehrheitlich auf Funktionen, die der Programmierer explizit aufruft, um sich hinsichtlich bestimmter Aspekte unterstützen zu lassen. Der Benutzer entscheidet selbst, wann er sprachbasierte Funktionen in Anspruch nimmt, und löst damit erst die syntaktische Analyse des aktuellen Codes aus; diese Prinzipien sind bereits im Editor *Z* [Wood 1981] Ende der 1970er Jahre implementiert. *Z* offerierte den Benutzern sowohl spezielle Unterstützung zum Programmieren als auch zum Bearbeiten von natürlichsprachlichem Text. So wurden verschiedene Strukturen berücksichtigt: Wörter (operationalisiert als Zeichenkette aus Buchstaben), Sätze (Zeichenfolgen, die mit einem Punkt enden) und Absätze (begrenzt durch Leerzeilen oder Spiegelstriche) [Wood 1981, S. 3]. Entlang dieser Strukturen sowie zeichen- und zeilenweise konnte beispielsweise der Cursor bewegt werden. Textelemente konnten kopiert und eingefügt sowie Strukturen wie Sätze und Absätze ausgerichtet werden [Whiteside et al. 1982, S. 4]. Sehr ähnlich dazu war auch das Konzept des Editors *ED3*, der konzipiert wurde als Universaleditor, d. h. für das Editieren von Text, Bildern und Daten. Auch hier wurden Texte als Abfolge bestimmter Strukturen gesehen. Diese Strukturen konnten bearbeitet werden und dienten gleichzeitig als Inhaltsverzeichnis des entsprechenden Textes [Strömfors und Jonesjö 1981, S. 22f].

Welche Funktionen dem Programmierer zur Verfügung stehen, ist abhängig von der aktuell verwendeten Programmiersprache. Viele Texteditoren unterstützen verschiedene Programmiersprachen in verschiedenen Modi, die entweder explizit vom Programmierer ausgewählt werden oder über die Dateiendung oder einen Kommentar in der ersten Zeile eines Programms automatisch erkannt werden. Funktionen basieren auf der syntaktischen Analyse des Codes. Texteditoren, die solche Funktionen zur Verfügung stellen – Van De Vanter et al. [1992, S. 441] nennen sie *syntax-recognizing editors* –, verwenden üblicherweise Werkzeuge zur syntaktischen Analyse, die prüfen, ob der aktuelle Code syntaktisch wohlgeformt ist. Der Programmierer muss sich lediglich mit semantischen Aspekten seines Codes beschäftigen.

In heutigen Texteditoren stehen Programmierern also Funktionen zur Verfügung, die die Elemente und Regeln der jeweiligen Programmiersprache berücksichtigen, *ohne* den Programmierer einzuschränken. Wissen über abstraktere Strukturen und Zusammenhänge (*higher-level knowledge*) wird benutzt, um Strukturen in Programmtexten zu erkennen und Funktionen darauf anwenden zu können. Damit kann der Programmierer auf abstrakteren Strukturen arbeiten und braucht sich nicht um die Umsetzung auf Zeichenebene zu kümmern.

Sprachbasierte Funktionen in Texteditoren verwenden die Elemente einer Programmiersprache – also die Schlüsselwörter – und die erlaubten bzw. geforderten Strukturen – also syntaktische Einheiten, wie Schleifen oder Blöcke. Die Tatsache, dass die einmalige Verwendung eines Variablennamens womöglich ein Fehler ist, wird von einem syntaktischen Prüfprogramm erkannt und dem Programmierer angezeigt. Allerdings wird nicht versucht, den Variablennamen zu korrigieren, da die Ursache dieses Fehlers nicht ermittelt werden kann: Es kann sich um einen Tippfehler handeln, das Programm ist vielleicht noch nicht fertig geschrieben oder diese Variable wird tatsächlich nicht weiter verwendet.

Formale Sprachen, wozu Programmiersprachen zählen, sind bezüglich ihres Lexikons und ihrer syntaktischen Regeln definiert und abgeschlossen. Das Lexikon ist in der Regel klein und enthält die Schlüsselwörter einer Programmiersprache. Es ist also relativ einfach, Parser zu implementieren, die im Code diese Schlüsselwörter erkennen: «Formal languages tend to be much simpler than natural ones since they are *designed* to be processed efficiently.» [Hess 1992, S. 130][2]. Anschliessend können diese Elemente beispielsweise optisch hervorgehoben werden – das sogenannte Syntaxhighlighting. Das Erkennen der syntaktischen Strukturen ermöglicht weiterhin die automatische Einrückung von Code-Zeilen, die Navigation auf diesen Strukturen und die Implementierung von Funktionen, die auf diesen Elementen und Strukturen operieren. Sprachbasierte Funktionen in Texteditoren profitieren also davon, dass die verwendete Sprache ein abgeschlossenes Inventar bezüglich Lexikon und Regeln hat.

4.1.2 Forschungslücke: Sprachbasierte Funktionen in Textbearbeitungsprogrammen

Hamlet [1986] vergleicht die Situation im Bereich Textbearbeitung Ende der 1980er Jahre mit der Situation bezüglich Editoren und Funktionen für Programmierer, *bevor* es Texteditoren mit sprachbewussten Funktionen gab. Er bezieht sich hauptsächlich auf die Kontrolle von «Text», wie sie in syntaxbasierten Editoren erzwungen wird, und schlägt vor, die Struktur eines zu schreibenden natürlichsprachlichen Textes ähnlich zu kontrollieren. Die Aussage:

> The analogy between writing and programming suggests that techniques and tools valuable in controlling software development be investigated for improving document preparation. [Hamlet 1986, S. 79]

können wir weiter fassen und generell auf heute in Texteditoren verfügbare Funktionen beziehen, die als sprachbasiert bezeichnet werden – also auf *syntax-recognizing editors*. Unseres Wissens ist dies die erste ausführliche Diskussion solcher Funktionen für die Bearbeitung natürlichsprachlichen Textes. Allerdings wurde die Idee nie weiterverfolgt und umgesetzt.[3]

Wie in Abschnitt 4.1.1 gezeigt, beruhen sprachbasierte Funktionen in Texteditoren auf bestimmten Eigenschaften formaler Sprachen. Natürliche Sprachen weisen ähnliche Elemente und Strukturen auf: Auch hier haben wir

2. Hervorhebung im Original
3. Persönliche Kommunikation mit Richard Hamlet, E-Mail vom 19. Juni 2006, Message-ID `<200606192000.k5JK01Lu020806@adara.cs.pdx.edu>`.

ein Lexikon und syntaktische Regeln. Das Lexikon natürlicher Sprachen ist jedoch nie abgeschlossen, da Wortbildungsprozesse, wie Komposition und Derivation, produktiv sind und jederzeit neue Wörter hinzukommen können. Die (morpho)syntaktischen Regeln sind in ihrer Anzahl zwar endlich, aber sehr viel umfangreicher als für Programmiersprachen. Diese Regeln sind zudem dem Sprachwandel unterworfen und können von der Sprachgemeinschaft prinzipiell jederzeit verändert werden.

Im Gegensatz zu formalen Sprachen *definieren* die Regeln natürlicher Sprachen nicht die Verwendung der Wörter, sondern *beschreiben* deren aktuelle Verwendung *ex post*. Hess schreibt: «The grammars of natural languages must be inferred laboriously from the language behaviour of native speakers, and we can never be sure that we have a correct, let alone complete, grammar available.» [Hess 1992, S. 131] Die Implementierung von Funktionen, die die Elemente und Strukturen von natürlichsprachlichem Text erkennen und deren Eigenschaften bestimmen, wird so erschwert. Die Grösse des Lexikons und die morphologischen Regeln einer Sprache führen dazu, dass für ein einzelnes Element eines Textes nicht jederzeit eine eindeutige Analyse geliefert werden kann – für Programmiersprachen sind keine Varianten von Schlüsselwörtern erlaubt, die Analyse ist nie ambig. Zudem ist in natürlichsprachlichen Texten die Verletzung syntaktischer Regeln aus stilistischen Gründen durchaus erlaubt, die Regeln sind nicht so strikt wie entsprechende syntaktische Regeln für Programmiersprachen.

Die einfache Übertragung von sprachbasierten Funktionen für Texteditoren auf Funktionen für Textbearbeitungsprogramme ist also nicht möglich. Auch wenn natürliche Sprachen nicht so eindeutig definierbar sind wie formale Sprachen, kann jedoch auf ihren Strukturen operiert werden. Die grundlegenden *Prinzipien* solcher Funktionen können wir also übertragen: das Erkennen von Elementen und Strukturen und die Operation auf diesen.

Wir finden in der Literatur nur sehr wenige Aussagen, die sich darauf beziehen, tatsächlich auf diesen Elementen zu operieren und dem Autor Funktionen zur Manipulation des bereits geschriebenen Textes anzubieten:

- Bereits Embley und Nagy [1981] wenden den Begriff *language-dependent editors* in einem Nebensatz auf das Bearbeiten von natürlichsprachlichen Texten an: «[…] language-dependent editors, which accomodate the syntactic rules of a programming language (or, eventually, even of natural language).» [Embley und Nagy 1981, S. 34], gehen jedoch nicht weiter darauf ein und beschäftigen sich lediglich mit Texteditoren.

- Kempen und Vosse [1992] gebrauchen den Begriff *language-sensitive*, beziehen sich jedoch ausdrücklich auf Prüf- und Korrekturprogramme. In ihrem Projekt wurden zwar auch Funktionen angekündigt, die es Autoren ermöglichen sollten, natürlichsprachliche Konstrukte zu manipulieren und Seiteneffekte automatisch zu korrigieren [Kempen et al. 1986, S. 370f] – die *Fünfte Generation* von Textbearbeitung nach «handwriting, printing, typewriting and word processing» [Kempen et al. 1986, S. 372]. Diese Funktionen wurden jedoch so nie implementiert [De Smedt und Kempen 1987]. Zudem war das Konzept sehr stark an syntaxbasierte Funktionen angelehnt, die den Autor zwingen, jederzeit syntaktisch wohlgeformten Text zu erstellen, ähnlich wie wir sie für Texteditoren in

Abschnitt 4.1.1 vorgestellt haben – eine Idee, die kritisiert wurde, sich als zur restriktiv erwiesen hat und aufgegeben wurde.

- Van De Vanter et al. [1992] verwenden den Begriff *language-based editing systems* und schliessen ausdrücklich das Editieren von natürlichen Sprachen im zu entwickelnden System *PAN* ein [Van De Vanter et al. 1992, S. 462f]. Allerdings wurden die Funktionen für natürliche Sprachen nie umgesetzt, es blieb bei einem Prototypen – das Fördergeld war aufgebraucht und die benötigte Infrastruktur nicht vorhanden [Van De Vanter und Boshernitsan 2000, S. 8].

- Im Zusammenhang mit der Analyse von Aufzeichnungen von Tastendrücken beim Redigieren von Texten stellen Severinson Eklundh und Kollberg [1996] fest, dass sich aus den einzelnen Tastendrücken die Redigierabsicht des Autors nicht erschliessen lässt. Dies sei dem Design der Textbearbeitungsprogramme geschuldet, die lediglich zeichenbasierte Funktionen besitzen. Sie schliessen daher:

  > If word processors could help writers edit their texts in *higher-level operations based on linguistic regularities of texts*, the editing pattern might be closer to the intended revising level and less character-oriented. [Severinson Eklundh und Kollberg 1996, S. 184][4]

- Dale [1997] spricht von *language sensitivity* und bezieht sich explizit auf Funktionen, die über das Finden, Diagnostizieren und Korrigieren von Fehlern hinausgehen:

  > The key here is to add intelligence and sophistication to provide *language sensitivity*, enabling the software to see a text not just as a sequence of characters, but as words and sentences combined in particular structures for particular semantic and pragmatic effect. [Dale 1997, S. 235][5]

  Dale äussert sich so in einer Vorhersage über die Entwicklung der nächsten fünf bis zehn Jahre und ordnet sie den *augmentativen* Technologien zu – wie wir in Kapitel 3 gezeigt haben, ist diese Entwicklung nicht eingetreten. Die 1997 erhältlichen Funktionen und Systeme zur Prüfung und Korrektur von orthographischen und grammatikalischen Fehlern bezeichnet er als «the current frontier of the state of the art» [Dale 1997, S. 236] in Bezug auf den Einsatz von computerlinguistischen Methoden und Ressourcen.

- Im Projekt «Integrated language tools for writing and document handling» der KTH Stockholm werden im Projektantrag ebenfalls linguistisch unterstützte Funktionen erwähnt:

  > These include linguistic search, i.e. searching for linguistic units rather than strings of characters. For example, a writer may need to locate all verbs in a text in order to consider the tense choice, and possibly change a verb to present instead of past tense. The latter is an example of linguistic editing

4. Hervorhebung hinzugefügt.
5. Hervorhebung im Original.

> functions, which use the linguistic structure of the text to provide powerful tools for revision.[6]

Jedoch wurde dieser Teil des Projektes nicht umgesetzt.[7] Im abschliessenden Bericht heisst es zum betreffenden Meilenstein: «A design specification for a linguistic editing function has not been written yet.»[8]

De Smedt [2009] gibt eine Zusammenfassung des Einsatzes von computerlinguistischen Ressourcen und Methoden, um das Schreiben zu erleichtern. Alle dort genannten Systeme stellen jedoch keine sprachbewussten Funktionen zur Verfügung, wie es Texteditoren tun. Die oben genannten Erwähnungen solcher Funktionen für Englisch, Niederländisch und Schwedisch wurden nie konzipiert oder gar implementiert.

Hier besteht also eine Forschungslücke, die wir mit unserem Konzept der linguistisch motivierten Funktionen in Editoren schliessen können. Im folgenden Abschnitt 4.2 stellen wir unseren Lösungsansatz vor.

4.2 Lösungsansatz

Wir verwenden sprachbasierte Funktionen in Texteditoren als Inspiration zur Konzeption und Implementierung entsprechender Funktionen zum Editieren von natürlichsprachlichen Texten. Aufgrund der in Abschnitt 4.1.2 dargestellten Unterschiede zwischen formalen und natürlichen Sprachen ist trotz prinzipieller Gemeinsamkeiten eine direkte Übertragung von Funktionen für Programmiersprachen auf natürliche Sprachen nicht möglich. Wir verwenden die *Prinzipien* von sprachbasierten Funktionen in Texteditoren: Sie beruhen auf dem Erkennen von bestimmten Elementen einer Sprache, deren Eigenschaften und relevanten Strukturen. Um die linguistischen Strukturen und Elemente eines natürlichsprachlichen Textes zu ermitteln und Funktionen zu implementieren, die auf diesen Strukturen operieren, ist der Einsatz von computerlinguistischen Ressourcen und Methoden notwendig.

Betrachten wir konkrete sprachbasierte Funktionen für Programmiersprachen, können wir drei *Typen* von Funktionen ermitteln, für die wir entsprechende Funktionen für natürliche Sprachen ableiten können. Sprachbasierte Funktionen allgemein können unterschieden werden in (a) Informationsfunktionen, (b) Bewegungsfunktionen und (c) Modifikationsfunktionen.

Informationsfunktionen Bereits Chusho et al. [1983] spezifizieren für den Editor *PARSE* Gruppen von Funktionen, etwa zum Abfragen von Informationen (bezeichnet als Funktionen im *Query mode*) – hinsichtlich intermodularer Abhängigkeiten, Parameterzugriffen, Skopus von Variablen und deren Beziehungen, Referenzierung von Variablen und Komplexität

6. Aus dem Projektbeschrieb, 18.9.1999, erhältlich im WWW `http://www.nada.kth.se/iplab/langtools/` (zuletzt besucht am 8.12.2010, 19:34).
7. Persönliche Auskunft der Projektleiterin Kerstin Severinson Eklundh, E-Mail vom 9. Juni 2008, Message-ID `<2BD74CC7-CB76-4412-BE34-AF89413C2245@csc.kth.se>`.
8. Siehe «Grammar checking and proof reading», erhältlich im WWW `http://www.csc.kth.se/tcs/projects/granska/rapporter/halfyear20000430.html` (zuletzt besucht am 8.12.2010, 19:34).

– und zum Modifizieren von Code (bezeichnet als *Reduction mode*). Für Letzteres unterscheiden sie lokale Modifikationen (wie das Einfügen, Löschen oder Verschieben von Strukturen) von globalen Modifikationen (wie das Umbenennen von Variablen oder das Verschmelzen oder Auftrennen von Schleifen und Prozeduren) [Chusho et al. 1983, S. 623].

Unter einer *hervorhebenden Informationsfunktion* verstehen wir eine Funktion, die bestimmte Elemente im Text explizit auszeichnet und keine weiteren Resultate an den Benutzer zurückliefert, entsprechend der Definition von Raskin:

> **Highlighting** is adding, by any means, a recognizable distinction to a displayed object. The function of the highlighting is to allow the user to determine, by passive observation, that the system has recognized a particular object as having a special status. The semantics of that status are dependent on the nature of the object […] [Raskin 2000, S. 105][9]

Die bekannteste Instantiierung einer solchen Informationsfunktion ist das *Syntaxhighlighting*. Schlüsselwörter, Variablennamen oder spezifische Strukturen der entsprechenden Programmiersprache werden farblich markiert, Strukturen einer Programmiersprache werden eingerückt (engl. *indentation*), um dem Programmierer in Sprachen wie Perl, C, Java oder Lisp die Orientierung im Code zu erleichtern. In Sprachen wie Python ist die Einrückung von Code-Elementen von syntaktischer Bedeutung, der Editor unterstützt den Programmierer, entsprechende Code-Elemente um die richtige Anzahl Leerzeichen einzurücken. Weiter ist es möglich, sich bei der Verwendung von Klammerstrukturen helfen zu lassen – der Editor zeigt an, wenn es zu viele schliessende oder öffnende Klammern gibt.

Die Objekte, die hervorgehoben werden können, stellen also sinntragende Einheiten dar. Raskin benennt zwar im Folgenden einzelne Zeichen als solche Elemente innerhalb eines Textes («In text, the object typically would be an individual character.» [Raskin 2000, S. 105]), dies ist jedoch den in Textbearbeitungsprogrammen vorhandenen Funktionen geschuldet, die auf Zeichenebene arbeiten. Einzelne Zeichen haben innerhalb eines natürlichsprachlichen Textes keine Bedeutung – lediglich auf Interpunktionszeichen träfe dies unter Umständen zu. Sinnvolle Einheiten eines natürlichsprachlichen Textes sind syntaktische Einheiten: Wörter, Wortgruppen, Satzglieder, Sätze.

Hervorhebende Informationsfunktionen machen also syntaktische Einheiten eines Textes deutlich sichtbar. Wir sind hier sehr viel näher an einem tatsächlichen syntaktischen Hervorheben, als es für Texteditoren zutrifft:

> "Syntax highlighting" is an unfortunate misnomer, since pattern-matching is considerably weaker than syntactic analysis. It would be more accurate to call it "unreliable keyword, string, and comment recognition". [Van De Vanter und Boshernitsan 2000, S. 3]

9. Hervorhebung im Original.

Zu hervorhebenden Informationsfunktionen zählen wir neben der Hervorhebung von Elementen, die bestimmte Eigenschaften aufweisen – etwa Verben in einer bestimmten Zeitform –, die Hervorhebung von syntaktischen Elementen, die bestimmte Eigenschaften *nicht* aufweisen – etwa Sätze ohne finites Verb.

Daneben definieren wir Informationsfunktionen, die explizit einen *Rückgabewert* liefern. Hier ist etwa zu denken an die Anzahl der Fundstellen eines Suchausdrucks, die verwendeten Wortformen eines Wortes, die Kategorie einer Wortform etc. Hervorhebende Informationsfunktionen können mit solchen, die einen Rückgabewert liefern, kombiniert werden – etwa das Hervorheben aller Fundstellen eines Suchausdrucks im Text mit der Angabe der Anzahl der Fundstellen.

Informationsfunktionen ohne linguistische Motivierung, die jedoch den Prozess des Redigierens bzw. allgemein den Schreibprozess als solchen als Basis verwenden, sind Funktionen, die etwa Strukturen oder Textabschnitte, die «zuletzt» intensiv bearbeitet wurden, hervorheben. Die Definition von «zuletzt» kann dabei durch den Autor selbst gewählt werden, etwa als «seit der letzten (automatischen) Speicherung des Textes», «alle Änderungen, die ein Koautor vorgenommen hat» oder «die letzten 20 Redigieroperationen». Werden Textstellen, die intensiv bearbeitet wurden, optisch hervorgehoben, wird es dem Autor erleichtert, Fehler selbst zu finden, die er beim normalen Korrekturlesen übersehen würde. Die benötigten Informationen hierfür sind eigentlich in allen Editoren bereits vorhanden – es ist möglich, mehrere Editieroperationen rückgängig zu machen. Yang hat exemplarisch in *GNU Emacs* gezeigt, wie die `undo`-Funktion zur Unterstützung des Redigierprozesses verwendet werden kann, siehe [Yang 1989, 1990, 1992].

Bewegungsfunktionen in Texteditoren (von Meyrowitz und van Dam [1982b, S. 354] *travelling operations* genannt) erlauben das Navigieren durch den Code entlang von Strukturen, also das Springen zum nächsten Schlüsselwort, zur nächsten Variableninstanz, zum Ende eines Blockes, zur Bedingungsangabe innerhalb der nächsten Schleife etc. Stehen solche Funktionen nicht zur Verfügung, ist das Navigieren sehr mühsam: «[…] cursor placement is the single most time consuming editing function next to typing itself. Thus, the mechanics of cursor positioning deserve considerable engineering attention and optimization.» [Whiteside et al. 1982, S. 38]

Die Strukturen, die für das Hervorheben in Informationsfunktionen verwendet werden, können als Ziele für Bewegungsfunktionen dienen. Für natürliche Sprachen können Bewegungsfunktionen also ermöglichen, zum Beginn des nächsten Satzes, zum nächsten oder vorigen finiten Verb zu navigieren.

Modifikationsfunktionen in Texteditoren erlauben die Manipulation der Elemente und Strukturen einer Programmiersprache, der *editing units*. So kann etwa ein Block komplett gelöscht oder verschoben werden, ohne ihn vorher explizit vom Beginn bis zum Ende zu markieren. Kommentare können gelöscht werden, Schleifen und Bedingungen werden automatisch ergänzt, syntaktische Elemente können ausgewählt und markiert werden. Wird eine Variable umbenannt, werden alle Verwendungen dieser Variablen umbenannt, Schlüsselwörter werden nach dem Eingeben

der Anfangsbuchstaben automatisch ergänzt etc. Einige Texteditoren stellen Modifikationsfunktionen zur Verfügung, die kontextsensitiv arbeiten: Für objektorientierte Programmiersprachen können so nur Methoden im Code verwendet werden, die das betreffende Objekt überhaupt zur Verfügung stellt.

Elemente und Strukturen, die manipuliert werden können, müssen zunächst markiert (engl. *select*) werden:

> **Selecting** is a process by which a user identifies a set of one or more objects as having a special status that can be recognized by the system, thereby creating a **selection** […] with the intention of applying a command to it in the near future. [Raskin 2000, S. 106]

Modifikationsfunktionen erlauben, auf den jeweiligen Elementen nur Manipulationen auszuführen, deren Resultat dazu führt, dass diese Elemente hinsichtlich der syntaktischen Regeln wohlgeformt sind – hier geht es also um die Verhinderung von Fehlern bei der Veränderung eines syntaktischen Elements. Jedoch berücksichtigen diese Funktionen nicht, ob in einem grösseren Kontext syntaktische Wohlgeformtheit vorliegt.

Übertragen wir dies auf natürlichsprachliche Texte, können beispielsweise Substantive in einen bestimmten Kasus gesetzt werden, aber nicht hinsichtlich Tempus oder Modus verändert werden, da sie entsprechende morphosyntaktische Eigenschaften nicht besitzen. Wird der Kasus eines Substantivs verändert, müssen allfällige Kongruenzverletzungen anschliessend korrigiert werden, damit die syntaktische Wohlgeformtheit im grösseren Kontext, d. h. im Satz, erhalten bleibt.

Wir können linguistisch unterstützte Funktionen zur Bearbeitung von natürlichsprachlichem Text in Anlehnung an sprachbasierte Funktionen in Texteditoren entwerfen und implementieren, also als *Informationsfunktionen*, *Bewegungsfunktionen* und *Modifikationsfunktionen*. Informationsfunktionen erleichtern es, einen Text hinsichtlich bestimmter Kriterien genauer zu betrachten, Bewegungsfunktionen ermöglichen schnelles zielgerichtetes Navigieren im Text und Modifikationsfunktionen erlauben die Manipulation von Elementen und Strukturen als solchen. Die drei Funktionstypen werden beim Redigieren kombiniert: Auf Grund der Resultate einer Informationsfunktion navigiert der Autor zu bestimmten syntaktischen Einheiten und nimmt dort eine Änderung vor.

Wir kommen noch einmal auf den Aspekt der Kreativität zurück, wie wir ihn in Abschnitt 2.3.1 bereits betont haben und wie er von Breidenbach im Hinblick auf das Redigieren beschrieben wird: «To be creative, however, revision needs time and freedom from excessive constraint and regimentation. It needs to remain open and loose and walk on the edge of possibilities, trying them on and checking them out.» [Breidenbach 2006, S. 200] Funktionen, die auf den Objekteinheiten einer Sprache – für natürlichsprachliche Texte also auf linguistischen Einheiten – operieren, erlauben es dem Autor, schnell Modifizierungen vorzunehmen, verschiedene Varianten auszuprobieren, Änderungen rückgängig zu machen. Der Autor muss dabei nicht alle kognitiven Ressourcen auf die korrekte Ausführung einer komplexen Sequenz kleinteiliger Funktionen verwenden, sondern kann sich auf sein eigentliches Redigierziel konzentrieren.

4.3 Klassifizierung sprachbasierter Funktionen

Bei der Erläuterung der drei Typen von sprachbasierten Funktionen im vorigen Abschnitt 4.2 haben wir einige Beispiele für solche Funktionen in Textbearbeitungsprogrammen gegeben. Für die Umsetzung unseres Konzeptes ist es notwendig, genau zu spezifizieren, welche Funktionen implementiert werden sollen, welche Resultate sie liefern, welche Ressourcen sie benötigen und wie sie vom Autor verwendet werden können. Für einen systematischen Überblick schlagen wir daher eine Taxonomie sprachbasierter Funktionen vor.

In der von uns vorgeschlagenen Taxonomie berücksichtigen wir Aspekte, die sich aus der Intention des Autors, der tatsächlichen Änderung im Text und dem Ausführen der erforderlichen Funktion im Textbearbeitungsprogramm ergeben. Diese Klassifizierung kann sowohl zur Spezifizierung entsprechender Funktionen in einem Textbearbeitungsprogramm als auch zur Beschreibung der vorgenommenen oder beabsichtigten Redigieroperationen verwendet werden. Wir nennen die Art der benötigten Ressourcen und gehen auf deren Komplexität ein. Jede Funktion kann anhand der Ausprägungen der folgenden Merkmale beschrieben werden.

Typ Wir unterscheiden *Informationsfunktionen*, *Bewegungsfunktionen* und *Modifikationsfunktionen*, siehe Abschnitt 4.2.

Informationsfunktionen markieren Elemente. Sie liefern dem Autor die Resultate bestimmter Analysen (*Informationsfunktionen mit Rückgabewert*) und können diese im Text hervorheben (*Hervorhebende Informationsfunktionen*). Der Text selbst wird nicht geändert.

Bewegungsfunktionen erlauben es, den Cursor schnell an eine bestimmte Position zu bewegen oder eine bestimmte Textstelle anzeigen zu lassen. *Nach* dem Ausführen einer Informationsfunktion oder einer Bewegungsfunktion nimmt der Autor möglicherweise Änderungen am Text vor, diese sind jedoch nicht notwendig mit der Bewegungsfunktion verbunden.

Modifikationsfunktionen verändern linguistische Elemente, d. h. einzelne Wörter, Wortgruppen, Satzglieder oder ganze Sätze.

Textveränderung Für das Kriterium *Textveränderung* kann ein boolescher Wert angegeben werden: Informations- und Bewegungsfunktionen ändern den Text nicht, Modifikationsfunktionen ändern den Text.

Spezifizität Einer Funktion können Argumente oder Parameter mitgegeben werden. Dies kann implizit – durch die Position des Cursors – oder explizit erfolgen. Eine Funktion kann sowohl implizite als auch explizite Argumente benutzen. Implizite Argumente müssen ermittelt werden, etwa das Wort und dessen Kategorie an der Cursorposition. Explizite Argumente müssen vom Benutzer eingegeben werden.

Spezifizität ist eine Eigenschaft für alle Funktionstypen.

Skopus Entsprechend der verwendeten Sprache wird angegeben, auf welchen strukturellen Elementen operiert wird oder für welche Elemente Informationen ermittelt werden sollen. Wir können hier auf die Taxonomien von Bridwell [1980], Faigley und Witte [1981], Sommers [1980] zurückgreifen

und benennen die linguistischen und textlinguistischen Elemente: Wort, Wortgruppe, Satzglied, Satz, Absatz, Abschnitt, Kapitel, Gesamttext. Beispielsweise ist der Skopus einer Modifikation zum konsistenten Ersetzen eines Wortes durch ein anderes die Wortebene.

Gebiet Eine Funktion kann lokal oder global sein. Eine Modifikationsfunktion zur Änderung der Reihenfolge der Elemente einer Koordination ändert den Text in einem sehr engen Gebiet um die aktuelle Cursor-Position und ist damit lokal. Diese Textstelle wählt der Autor bewusst aus, sie ist auf dem Bildschirm sichtbar. Eine Modifikationsfunktion zur Ersetzung eines Wortes durch ein anderes im gesamten Text soll jedoch Auswirkungen auf alle Textstellen haben, die durch die Angabe des zu ersetzenden Wortes ermittelt werden können. Damit ist diese Funktion inhärent global, selbst wenn schliesslich nur eine Textstelle betroffen sein mag. Dabei muss der Autor weder beim Aufruf noch während der Ausführung der Funktion die betroffenen Textstellen notwendigerweise auf dem Bildschirm sehen.

Das Gebiet der Gültigkeit kann sowohl für Modifikationsfunktionen als auch für Informations- und Bewegungsfunktionen angegeben werden.

Linguistische Ressourcen Informations-, Bewegungs- und Modifikationsfunktionen können computerlinguistische Ressourcen verwenden. Für einige Funktionen wie das Vertauschen der Elemente einer Koordinaton ist es ausreichend, die grundlegenden Funktionen eines Textbearbeitungsprogamms (also `cut`, `copy`, `paste` etc.) unter Berücksichtigung der Eigenschaften des Schreibsystems einer bestimmten Sprache geschickt zu kombinieren – hier sind keine weiteren Ressourcen notwendig. Es ist nicht nötig, die beteiligten Wortformen morphologisch zu analysieren.

Informationsfunktionen zur Hervorhebung von Wortformen einer bestimmten Wortart benötigen linguistische Ressourcen. Wird lediglich mit nichtflektierenden Wortarten gearbeitet, mögen statische Ressourcen wie Listen genügen. Der Einbezug von morphologischen oder syntaktischen Regeln ist hier nicht notwendig.

Funktionen, die auf flektierenden Wortarten operieren, benötigen dynamische computerlinguistische Ressourcen, also Methoden und Systeme, die zur Laufzeit der Funktion ausgeführt werden. Modifikationsfunktionen zum Ersetzen von Wörtern oder Verändern von Wortgruppen verwenden sowohl morphologische Analyse als auch Generierung. Möglicherweise ist auch syntaktische Analyse notwendig.

Für dynamische Ressourcen können wir unterscheiden, ob sie für die *Analyse* – also das Bestimmen bestimmter Eigenschaften bereits geschriebener Wörter, Wortgruppen etc. – oder für die *Generierung* eingesetzt werden – also für die Erzeugung bestimmter linguistischer Elemente. Zudem kann der Skopus der Ressource angegeben werden entsprechend der Art der behandelten linguistischen Elemente: Wörter, Wortgruppen, Satzglieder, Sätze. Hier wird angegeben, ob die Ressourcen auf *morphologischer* oder *syntaktischer* Ebene eingesetzt werden.

Die benötigte linguistische Ressource wird also durch die Ausprägung von drei Eigenschaften beschrieben: (a) statisch vs. dynamisch, (b) für die Analyse vs. für die Generierung, (c) morphologisch vs. syntaktisch. Wir können dies als Vektor darstellen. Eine statische Ressource hätte den Wert:

$$\begin{pmatrix}1\\0\\0\end{pmatrix}$$

Eine Ressource zur automatischen morphologischen Analyse ist dynamisch und hat den Wert:

$$\begin{pmatrix}2\\1\\1\end{pmatrix}$$

Für eine Ressource, die sowohl zur automatischen morphologischen Analyse als auch zur automatischen morphologischen Generierung verwendet werden kann, addieren sich die beiden Vektoren:

$$\begin{pmatrix}2\\1\\1\end{pmatrix} + \begin{pmatrix}2\\2\\1\end{pmatrix} = \begin{pmatrix}4\\3\\2\end{pmatrix}$$

Wird keine linguistische Ressource verwendet, ist der Wert:

$$\begin{pmatrix}0\\0\\0\end{pmatrix}$$

Es ist möglich, dass der Skopus einer Funktion eine Wortgruppe ist und diese Funktion Ressourcen mit dem Skopus Wort zur Analyse und Generierung verwendet.

Seiteneffekte Als Seiteneffekte werden üblicherweise nichtintendierte Auswirkungen einer Operation bezeichnet: «A **side effect** is an effect of a command that alters contents or events that are not your locus of attention.» [Raskin 2000, S. 113][10] Wir erweitern diese Definition hier wie folgt:

Wird eine Modifikationsfunktion wie `toggle-number` für eine Substantivgruppe ausgeführt, ist das Resultat grammatisch, d. h., die beteiligten Wortformen der Substantivgruppe sind kongruent. Bezieht man den Kontext des behandelten Elements mit ein, z. B. das finite Verb des Satzes, ist die Textstelle eventuell nicht mehr wohlgeformt: Wurde das Subjekt pluralisiert, sind Subjekt und finites Verb nicht mehr kongruent. Das finite Verb wurde nicht unbeabsichtigt geändert oder gelöscht, wie es die ursprüngliche Definition für Seiteneffekt beschreibt, sondern die syntaktischen Beziehungen zwischen den geänderten Wortformen und den unveränderten vorhandenen Wortformen haben sich geändert. Dies ist klar ein unbeabsichtigter Seiteneffekt. Es sind weitere Redigieroperationen nötig, um diese Auswirkung zu korrigieren.

Die Reichweite solcher Seiteneffekte kann sehr gross sein. Seiteneffekte können (a) innerhalb einer Wortgruppe auftreten – z. B. bezüglich Adjektiv-Substantiv-Kongruenz durch die Ersetzung eines Substantives durch ein anderes –, (b) innerhalb eines Satzes – z. B. bezüglich Subjekt-Verb-Kongruenz wie beschrieben – oder (c) sogar über Satzgrenzen hinaus, wenn anaphorische Referenzen betroffen sind. Seiteneffekte können jedoch nur auftreten, wenn entsprechende Satzteile bereits geschrieben sind. Wird also eine Substantivgruppe pluralisiert, das Verb und weitere Satzglieder sind jedoch noch nicht geschrieben, treten keine Seiteneffekte im Text auf.

10. Hervorhebung im Original.

Wir bestimmen also, ob eine Funktion Seiteneffekte haben kann, und können diese dann klassifizieren. Seiteneffekte können nur für Modifikationsfunktionen, nicht für Informations- und Bewegungsfunktionen auftreten, da letztere den Text nicht manipulieren.

Modus Abhängig davon, ob während des Ausführens einer bestimmten Funktion andere Funktionen ausgeführt werden dürfen, ist es notwendig, diese Funktion in einem bestimmten Modus aufzurufen, um etwa das Ausführen anderer Funktionen oder die Eingabe von Text zu verhindern. Für Informations- und Bewegungsfunktionen, die den Text nicht verändern, ist kein eigener Modus erforderlich. Diese Funktionen müssen in der Regel auch nicht durch den Autor explizit beendet werden, sondern sind nach der Ausgabe oder Anzeige der gewünschten Information bzw. nach dem Abschluss der Bewegung des Cursors abgeschlossen. Für Funktionen, die die Interaktion mit dem Benutzer erfordern oder ermöglichen, kann es dagegen notwendig sein, andere Funktionen bis zum Abschluss der Funktion zu deaktivieren.

Wir unterscheiden also, ob eine Funktion in einem eigenen Modus ausgeführt werden sollte, d. h., der Wert für dieses Kriterium ist ebenfalls ein boolescher Wert.

Diese Klassifizierung verwenden wir in Kapitel 7, um die dort vorgestellten Funktionen jeweils zu kategorisieren. Sie erleichtert die Spezifikation der Funktionen und dient als Richtlinie der aktuellen Implementierung. Wir erhalten damit neben der natürlichsprachlichen Beschreibung einer Funktion Angaben zu deren Eigenschaften. Daraus lassen sich Funktionsfamilien ableiten. Die Eigenschaften einer Funktion oder einer Funktionsfamilie ermöglichen es weiterhin, Anforderungen an die Implementierung zu spezifizieren oder sogar Pseudocode zu erhalten. Aus der Funktionsmatrix wählen wir prototypische Funktionen aus, die das Spektrum der verschiedenen Elemente der Klassifizierung abdecken und die wir in Kapitel 7 detailliert diskutieren.

Die hier vorgeschlagene Klassifizierung fokussiert auf die programmiertechnische Umsetzung der Operationen, die ein Autor ausführt. Die so erzeugten Änderungen des Textes können dann wiederum in die bekannten Taxonomien von Bridwell [1980], Faigley und Witte [1981], Lindgren und Sullivan [2006] und Sommers [1980] eingeordnet werden.

Im nächsten Abschnitt 4.4 erörtern wir, welchen Prinzipien eine Implementierung der so beschreibbaren Funktionen folgen muss.

4.4 Leitlinien für die Implementierung

Die Umsetzung des Konzepts als Implementierung konkreter Funktionen folgt verschiedenen Vorgaben, die wir in den folgenden Abschnitten erläutern. Wir gehen in Abschnitt 4.4.1 auf Designprinzipien ein, die wir aus der Implementierung von sprachbasierten Funktionen für Texteditoren ableiten können. In Abschnitt 4.4.2 zeigen wir, dass Funktionen in Editoren allgemein gültigen Prinzipien der Bedienbarkeit und Gebrauchstauglichkeit folgen müssen und gehen in Abschnitt 4.4.3 auf die Ressourcen ein, die wir verwenden.

4.4.1 Designprinzipien

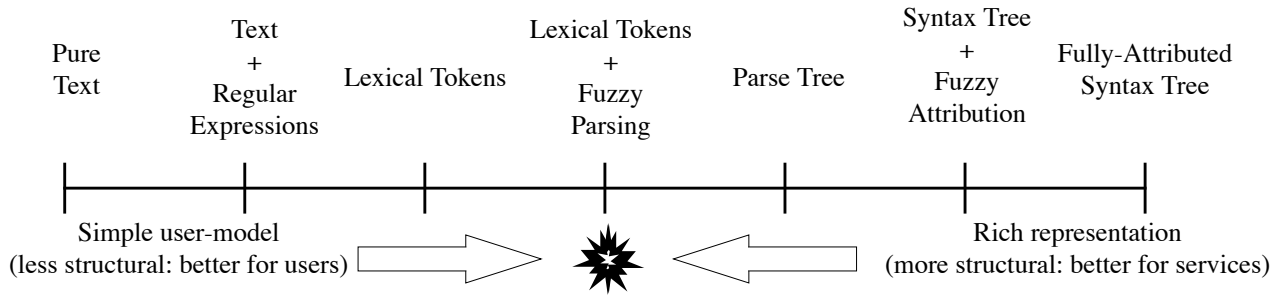


Figure 2: Additional choices for program representation and analysis

Abbildung 4.1: Design von sprachbasierten Funktionen in Texteditoren von Van De Vanter und Boshernitsan [2000, S. 4]

Sprachbasierte Funktionen in aktuellen Texteditoren verwenden Methoden und Ressourcen, um den bereits geschriebenen Text zu analysieren und Operationen auf den ermittelten Einheiten zu erlauben. Je detaillierter und genauer die Analyse ist, desto bessere Funktionen können implementiert werden. Die Implementierung elaborierter Funktionen ist also abhängig von der Ermittlung bestimmter Einheiten und deren Eigenschaften. Texteinheiten (etwa Blöcke in Programmcode oder Sätze in natürlichsprachlichem Text) sind jedoch während des Schreibens unvollständig und enthalten oft syntaktische Fehler. Van De Vanter bezeichnet dies in sprachbasierten Texteditoren als die «drei I»: *Ill-Formedness*, *Incompleteness* und *Inconsistency*, welche entsprechende Funktionen berücksichtigen müssen [Van De Vanter 1995, S. 255f]. Eine tiefe Analyse solcher Einheiten ist nicht möglich, vielmehr geht es darum, einen Kompromiss zu finden. «[…] where the amount of language analysis performed is as simple (and localized) as possible, but also as useful as possible.» [Van De Vanter und Boshernitsan 2000, S. 4] Van De Vanter und Boshernitsan bezeichnen dies als additive choices, verorten diesen Ansatz in Abbildung 4.1 und zeigen die Bestimmung lexikalischer Einheiten (also mehr als das) und der Bestimmung syntaktischer Beziehungen (also weniger als das) dieser Einheiten.

Auch diesen Aspekt können wir auf sprachbasierte Funktionen in Textbearbeitungsprogrammen übertragen. Linguistisch unterstützte Funktionen können verschiedene computerlinguistische Methoden und Ressourcen verwenden, um Strukturen im bereits geschriebenen Text zu erkennen und zu analysieren. Solche Ressourcen können prinzipiell einfache Analysen (etwa die automatische Worterkennung) oder komplexe linguistische Analysen liefern. Wir finden also eine ähnliche Bandbreite vor wie für die Analyse von Programmcode entsprechend Abbildung 4.1.

Wir betrachten Redigieren als Phase innerhalb des Schreibprozesses, die nicht am Ende des Schreibprozesses auftritt, sondern mit anderen Phasen des Planens und Umsetzens iterativ und rekursiv verwoben ist, also als *revising-while-composing* nach Sharples [1999, S. 105ff]. Autoren lesen, evaluieren und ändern Text, *bevor* der aktuelle geschriebene Satz beendet, also vollständig ist. Dies belegen Studien, die sowohl die Entwicklung des Textes mittels Analyse der Tastendrücke als auch den Fokus des Autors mittels Eye-Tracking untersuchen, siehe [Johansson et al. 2008, Van Waes et al. 2010, Van Waes und Schellens 2003] oder [Wengelin et al. 2009]. Die redigierten Teilsätze sind also unvollständig,

- Thus the lexical representation not used by any prior systems emerges as a promising compromise:
- It is a stream, not a tree, and thus bears a close relationship to textual source code;
- The analysis needed to update the representation after each edit usually requires only local context;
- It has enough linguistic information to provide many language-based services, including more robust implementation of familiar services such as identation, parenthesis and bracket matching, procedure/method head recognition, etc.; and
- It is a language representation suitable for integration with other tools, including complete language analyzers. Further analysis, for example parsing, could be folded into the CodeProcessor if added carefully, but at some additional cost in complexity.

Although this approach is promising, a number of design questions remain:

- Can the textual display and behavior be made to look and feel familiar enough that it requires no training?
- To what degree can the display be specialized for programs using only lexical information?
- Can such a fine-grained typographical display be implemented using current toolkit technology and made configurable?
- Can the lexical token representation be made robust in the presence of partially typed and therefore syntactically incorrect tokens? In particular, how can "bracketed" tokens such as string literals be managed when one of the brackets (double quotes for strings) is missing?
- What specialized support for comments and other, possibly non-textual, annotations is possible?
- How can a description-driven lexical analyzer be adapted to update the representation after each keystroke?

Solutions appear in the following two sections, which summarize respectively the two mutually dependent aspects of the CodeProcessor design architecture/implementation and user model. The architecture is presented first in Section 5, although many aspects were driven by the user-model design described in Section 6.

inkonsistent und nicht wohlgeformt und entsprechen den «drei I» von Van De Vanter [1995].

Es werden Funktionen benötigt, die nicht von Ressourcen abhängig sind, die auf der grammatischen Wohlgeformtheit des bereits Geschriebenen beruhen. Eine tiefe syntaktische Analyse des bereits Geschriebenen ist nicht ohne Weiteres möglich. Wenn heutige Hardware auch erlauben würde, eine solche Analyse zu jedem beliebigen Zeitpunkt schnell genug durchzuführen, um sie in interaktiven Funktionen verwenden zu können, sind aktuell verfügbare syntaktische Parser für Deutsch nicht robust genug, korrekte detaillierte Analysen für unvollständige Sätze zu produzieren.

Wir richten uns klar am Bedürfnis aus, die kognitiven Anforderungen beim Redigieren zu reduzieren und den Autor von unnötigen Zwischenschritten zur Erreichung seines Redigierziels zu entlasten. Der Autor kontrolliert, was mit dem Text geschieht, nicht das System. Computerlinguistisch interessante Operationen, die jedoch nicht zuverlässig korrekt automatisch ausgeführt werden können, schliessen wir daher aus – die Korrektur der Ergebnisse solcher Operationen erfordert zusätzliche Arbeit vom Autor. Beispielsweise mag es sich aufdrängen, nur Operationen zuzulassen, deren Ergebnis eine wohlgeformte Struktur ist *und* die in einen anschliessend auch wohlgeformten Satz eingebettet ist. Wir können jedoch den Autor nicht zwingen, jeweils erst einen Satz vollständig zu schreiben, bevor wir Operationen auf Teilen des Satzes erlauben – wie wir in Abschnitt 2.3 gezeigt haben, redigieren Autoren bereits, bevor der aktuelle Satz vollständig geschrieben ist, unabhängig vom verwendeten Schreibwerkzeug. Zudem können wir nicht vorhersagen, was der Autor mit einer modifizierten Struktur anschliessend beabsichtigt. Werden weitere Redigierschritte ausgeführt, ist die Anpassung des Kontextes unnötiger Aufwand.

Ein Autor soll von eintönigen und gleichzeitig oder sogar deswegen fehleranfälligen Aufgaben entlastet werden, um mehr Kapazität für kreative, anspruchsvolle Entscheidungen zu haben. Die entsprechend grammatischen Regeln beste Lösung ist unter Umständen nicht adäquat für ein bestimmtes rhetorisches Ziel eines Autors. Darum sollte ihm keine Lösung präsentiert werden, die den Anspruch erhebt, perfekt zu sein – gegen eine solche Beurteilung einer Maschine zu entscheiden und diese Lösung zu ändern, erfordert wiederum kognitive Kapazität. Vielmehr folgen wir der Anmerkung von Hamlet [1986, S. 88]: «This is the proper use of an imperfect rule-driven program: its output is expected to be examined by a person.» Kann keine eindeutige Lösung ermittelt werden, entscheidet der Autor sich für eine der präsentierten Varianten.

Heutige Hardware ist sehr viel leistungsfähiger als vor 30, 20 oder 10 Jahren. Selbst rechenintensive Prozesse können heute in sehr kurzer Zeit auf handelsüblichen Laptops durchgeführt werden.[11] Damit können notwendige Prozesse, basierend auf computerlinguistischen Systemen und Ressourcen, in einer Geschwindigkeit durchgeführt werden, die tatsächlich interaktive Funktionen ermöglicht. Benutzer erwarten Rückmeldungen des Systems bzw. Ergebnisse innerhalb sehr kurzer Zeit. Insbesondere für Operationen, die vom Benutzer als «einfach» eingeschätzt werden, werden lange Wartezeiten nicht toleriert:

11. Zum Vergleich: Für die Erstellung der 168'392 Einträge des Allomorphlexikons aus 98'545 Grundformeinträgen für die *Spanische Malaga Morphologie* [Mahlow 2000] wurden im Jahr 2000 auf einer HP B132L mehrere Stunden benötigt; auf einem MacBook (2.16 GHz Intel Core 2 Duo processor, 2 GB RAM mit Mac OS X 10.5.4) dauert dies 9 Sekunden.

Akzeptiert werden Reaktionszeiten zwischen 0,1 und 2 Sekunden [Cooper et al. 2007], für «komplexe» Funktionen werden grössere Wartezeiten bis 15 Sekunden toleriert [Good 1981, S. 40]. Ist die Wartezeit grösser als angenommen, wird dies als negativ wahrgenommen:

> The duration of the machine response time, *R*, can have an unexpected effect on user actions. If a user operates a control and nothing appears on the display for more than approximately 250 msec, she is likely to become uneasy, to try again, or to begin to wonder whether the system is failing. [Raskin 2000, S. 75]

Projekte, die linguistische Ressourcen in Editoren integrieren, um Autoren etwa Prüf- und Korrekturfunktionen anzubieten, gingen bislang davon aus, diese in einen eigens zu entwickelnden Editor einzubinden, siehe Abschnitt 3.2. Natürlich gab es in den 1970er und 1980er Jahren noch nicht sehr viele Textbearbeitungsprogramme und vor allem waren diese nicht weit verbreitet, Benutzer waren daher eher bereit, sich auf einen neuen Editor einzustellen. Spätestens jedoch mit der Etablierung bestimmter Editoren mit bestimmten Funktionen nahm die Akzeptanz von immer neuen Prototypen, die zwar einige ausgeklügelte Funktionen anboten, jedoch Kernfunktionen nicht enthielten, radikal ab [vgl. Van De Vanter 1992, Van De Vanter und Boshernitsan 2000]. Linguistisch basierte Funktionen sollen alltägliches Schreiben erleichtern. Es kann daher nur darum gehen, Autoren *zusätzliche* Funktionen anzubieten, die mit den bisher verwendeten nicht interferieren und die ihnen das Verwenden ihres bisher favorisierten Editors ermöglichen.

Sprachbasierte Funktionen – ob in Texteditoren oder in Textbearbeitungsprogrammen – beruhen auf dem Prinzip, dass der Autor jeweils entscheidet, welche Aktion er vornehmen möchte. Die Funktion erleichtert ihm die Ausführung, versucht jedoch nicht, weitergehende Absichten des Autors zu «erraten». Alle Funktionen in einem Editor sollten sich ähnlich bedienen lassen; der Benutzer sollte in der Lage sein, für eine neue Funktion deren Verhalten vorherzusagen. Funktionen, die darauf beruhen, dass nur der Autor selbst die Kontrolle über den Text hat, können also nur in einen Editor integriert werden, der bereits diesem Prinzip folgt. In Abschnitt 7.1 gehen wir genauer darauf ein.

Die Qualität der Resultate einer konkreten Funktion ist stark abhängig von der Qualität der verwendeten Ressourcen: Werden lediglich bereits vorhandene Kernfunktionen eines Editors geschickt kombiniert, beeinflusst die Art dieser Kombination die Qualität. Werden computerlinguistische Ressourcen einbezogen, hängt die Qualität der gesamten Funktion zusätzlich von der Qualität dieser Ressourcen und ihrer Kombination ab. Computerlinguistische Ressourcen werden kontinuierlich weiterentwickelt und es werden gänzlich neue Ressourcen entwickelt.[12]

Wie zufrieden ein Benutzer mit einer bestimmten Funktion ist, hängt zudem von der Art der Bedienbarkeit ab: Wie schwierig ist es, die Funktion aufzurufen und ihr die benötigten Informationen oder Parameter mitzugeben, wie aussagekräftig sind die zurückgelieferten Resultate, wie gut kann man mit diesen weiterarbeiten. Diese Aspekte betrachten wir im folgenden Abschnitt 4.4.2

12. Eine Ressource ist «nicht verfügbar», wenn die entsprechende Lizenz ihre Verwendung nicht erlaubt oder eine benötigte Ressource tatsächlich (noch) nicht existiert.

unter dem Stichwort *usability* im Sinne von *Gebrauchstauglichkeit* und *Bedienfreundlichkeit*.

4.4.2 Usability

Die Kriterien für Gebrauchstauglichkeit sind in der Norm ISO 9241-11 [ISO 1998] definiert: Effektivität, Effizienz und Zufriedenstellung. Gebrauchstauglichkeit bezieht sich hauptsächlich auf die Qualität von Software als solche; Bedienfreundlichkeit bezieht sich dagegen auf Qualitätskriterien zur *Verwendung* von Software oder Funktionen bzw. darauf, wie einfach Funktionen zu bedienen sind: Erlernbarkeit, Erinnerbarkeit, Effizienz, Fehleranfälligkeit und Zufriedenheit der Benutzer. [vgl. Nielsen 1993]

Werden Funktionen zu einem bereits existierenden Editor hinzugefügt, sollten diese ähnlich den bereits existierenden Funktionen aufgerufen werden können, um Erlernbarkeit und Erinnerbarkeit zu unterstützen.

Entsprechend verschiedenen Schreib- und Redigierstile überlappen sich Schreiben und Redigieren oder sind klar voneinander abgrenzbar, siehe [Galbraith und Torrance 2004, Lutz 1983, Rijlaarsdam et al. 2004] und [Severinson Eklundh 1994]. Für den letzten Fall, in dem explizit und bewusst eine Redigierphase durch den Schreiber begonnen wird, ist vorstellbar, in einen entsprechenden Modus zu wechseln. Denn nur in der Redigierphase sind Informationsfunktionen mit Rückgabewerten sinnvoll. Modifikationsfunktionen, Bewegungsfunktionen und hervorhebende Informationsfunktionen können sowohl in expliziten Schreibphasen als auch in Mischformen und in expliziten Redigierphasen eingesetzt werden.

Die Fehlerrate von Funktionen, die computerlinguistische Ressourcen verwenden, ist teilweise voraussagbar (siehe Kapitel 6), sodass die Notwendigkeit der Interaktion mit dem Benutzer ebenfalls vorhergesehen werden kann. Für Funktionen, die auf mehreren Ressourcen oder gar auf einer Kombination solcher Ressourcen beruhen, muss ein eigenes Bedienkonzept verwendet werden oder sogar aufgrund der Qualität der heute verfügbaren linguistischen Ressourcen momentan auf die Implementierung verzichtet werden – dies gilt beispielsweise für Funktionen, die Anaphernresolution erfordern würden. Vor allem für globale Modifikationsfunktionen ist eine Verwendung in einem expliziten Redigiermodus sinnvoll, in welchem sich der Benutzer bewusst ist, dass er stark interagieren muss – aus angebotenen Varianten auswählen, Parameter nachträglich spezifizieren, markierte Textstellen anschliessend selbst noch einmal kontrollieren und gegebenenfalls ändern etc.

Wichtig ist, dass eine Funktion einfach zu bedienen ist, also durch *eine kurze* Tastenkombination, *einen* Knopf oder *einen* Menü-Aufruf [Allen und Scerbo 1983, S. 176f]. Kann eine Editieroperation durch die Kombination schon vorhandener Funktionen umgesetzt werden (etwa das Vertauschen der Elemente einer Koordination), sollte ein Makro dafür erstellt und verwendet werden. In der Regel werden solche Funktionen sogar relativ sprachunabhängig sein. Sie benutzen die vorhandenen Konzepte des Editors und kombinieren sie geschickt, um eine Funktionalität zu erreichen, die auf einer neuen Ebene operiert [Ballance et al. 1992, S. 123].

Die Funktionen sollen den Autor nicht einschränken und ihn nicht von der Umsetzung einer Idee abhalten, sondern ihn bei deren Realisierung unterstützen. Die Funktionen müssen sich in der Art, wie sie aufgerufen werden und wie Rückmeldungen gegeben werden, an die Vorlieben des Benutzers anpassen lassen: «We must design interfaces that (a) deliberately take advantage of the human trait of habit development and (b) allow users to develop habits that smooth the flow of their work. » [Raskin 2000, S. 20]

Insgesamt sollen die vorgeschlagenen linguistisch basierten Redigierfunktionen also auch den Anforderungen entsprechen, die in der Norm ISO/IEC 9126–1 [ISO 2001] definiert werden: Programme sollen funktional sein, zuverlässig funktionieren, einfach zu benutzen sein, sich effizient verhalten, einfach zu warten sowie übertragbar sein.

4.4.3 Einsatz computerlinguistischer Ressourcen

Unser Ziel ist die Vermeidung typischer Redigierfehler und die Reduzierung der kognitiven Beanspruchung des Benutzers, d. h. des Autors. Der Autor soll sich darauf konzentrieren können, sein kommunikatives Ziel zu erreichen. Alle Aufgaben, die mit dem eigentlichen Ziel nicht zusammenhängen, sollen vermieden werden. Da wir auf den linguistischen Einheiten eines Textes arbeiten wollen, müssen wir diese Einheiten erkennen. Es handelt sich um *Wort*, *Wortgruppe*, *Satzglied*, *Satz*. Für die meisten dieser Einheiten müssen wir zudem auf die relevanten Eigenschaften zugreifen können, um eine bestimmte Instanz zu manipulieren oder um zu entscheiden, ob sie als Objekt einer Funktion in Frage kommt. Wir benötigen also Zugriff auf die morphosyntaktischen Eigenschaften der linguistischen Einheiten.

In der Computerlinguistik haben sich verschiedene Klassen von Systemen zur Bestimmung solcher Strukturen und ihrer Eigenschaften etabliert: (a) automatische Wortartenbestimmer und Systeme zur morphologischen Analyse für Wörter, (b) Systeme zur syntaktischen Analyse für Wortgruppen, Satzglieder und Sätze. Um diese Einheiten tatsächlich manipulieren zu können, werden entsprechende Systeme zur Generierung verwendet, ebenfalls für Wörter, Wortgruppen, Satzglieder und Sätze.

Es sollen möglichst keine neuen Aufgaben an den Autor delegiert werden, etwa erzwungene Interaktion, um morphologische oder syntaktische Kategorien zu bestimmen, zu korrigieren oder aus einer Liste auszuwählen. Wir benötigen daher Ressourcen, die möglichst korrekte Resultate liefern, Ambiguitäten sollten aufgelöst werden. Meurers et al. argumentieren, dass ein Korrektheitsgrad von mehr als 90 % für eine computerlinguistische Ressource notwendig ist, um in einer praxistauglichen Anwendung verwendet zu werden. Es ist weniger tragisch, wenn beispielsweise nicht alle Vorkommen einer Wortart gefunden werden, die gefundenen Wortformen müssen jedoch möglichst korrekt identifiziert worden sein, um den Benutzer nicht zu verwirren. Hinsichtlich Ausbeute (*recall*) können also relativ niedrige Werte in Kauf genommen werden, Präzision (*precision*) sollte jedoch etwa 95 % (siehe [Meurers et al. 2010, S. 15] und persönliche Kommunikation mit Detmar Meurers), bevorzugt jedoch 100 % [Hull et al. 1987] erreichen. Das üblicherweise verwendete f-Mass (f-*Measure*) [van Rijsbergen 1979] ist hier also nicht aussagekräftig genug.

Für komplexe Aufgaben müssen verschiedene dieser Ressourcen kombiniert werden. Dabei ist zu berücksichtigen, dass sich Fehler in einzelnen Systemen bei der Kombination extrem negativ auswirken können. Computerlinguistische Ressourcen können heute schnell aufgerufen und ausgeführt werden, jedoch erreichen sie nie 100 % Korrektheit hinsichtlich Ausbeute und Präzision. Zudem kann nicht vorhergesagt werden, *welche* Ergebnisse vermutlich inkorrekt oder unvollständig sind. Werden solche Ressourcen in Funktionen integriert, muss der Benutzer sich bewusst sein, dass Fehler auftreten können. In Kapitel 6 stellen wir dar, welche Ressourcen wir verwenden. In Abschnitt 7.2 gehen wir auf die konkrete Einbindung dieser Ressourcen ein.

Welche der prinzipiell möglichen und anhand der Taxonomie beschreibbaren Funktionen sind es tatsächlich wert, implementiert zu werden? Natürlich hängt dies einerseits davon ab, welche computerlinguistischen Ressourcen benötigt werden und wie gut deren Qualität ist. Dies wiederum ist abhängig vom Entwicklungsstand der entsprechenden Ressourcenart und von den Eigenschaften der Sprache, hier also Deutsch. Selbst wenn die Ressource immer korrekte Ergebnisse liefert, können diese ambig sein und Interaktion mit dem Autor erfordern. Beispielsweise kann der Nominativ Plural des Wortes «*Mutter*» sowohl «*Mütter*» als auch «*Muttern*» sein. Da wir Funktionen implementieren wollen, die die kognitive Beanspruchung des Autors reduzieren, ist die Auflösung solcher Ambiguitäten durch Interaktion mit dem Autor nicht immer eine gute Idee. Hier ist dann zu entscheiden, ob die Implementierung einer Funktion überhaupt sinnvoll ist.

Ein weiteres Kriterium ist die tatsächliche Fehleranfälligkeit bestimmter Redigieroperationen. Diese korreliert vermutlich mit der Komplexität einer entsprechenden Editierfunktion. Die Entwickler von *RUSKIN* stellten erst am Ende des Projektes fest, dass die Autoren ganz andere Funktionen gewünscht und gebraucht hätten:

> Yet only towards the end of this work did we realize that we had approached the problem the wrong way arround. We had not asked writers what they would like software to do for them; nor conducted an analysis of the critical problems in writing and considered how computers could help. [Williams 1990b, S. 6]

Darum sollte sich die Entscheidung, welche Funktionen wir implementieren, auf empirische Befunde der Schreibforschung stützen, wie auch Sharples [1996b] fordert. Wir werden dies in Kapitel 5 erörtern.

4.5 Zusammenfassung

In diesem Kapitel haben wir das Konzept linguistisch unterstützter Redigierfunktionen in Textbearbeitungsprogrammen entwickelt. Ausgehend von entsprechenden Funktionen in Texteditoren können wir sprachbasierte Funktionen als Informationsfunktionen, Bewegungsfunktionen und Modifikationsfunktionen klassifizieren. Diese Funktionstypen können wir auf die Bearbeitung von natürlichsprachlichem Text übertragen, da natürliche und formale Sprachen Gemeinsamkeiten aufweisen: Sie können mittels eines Lexikons und grammatischer Regeln beschrieben werden. Das Lexikon natürlicher Sprachen ist grösser

und die Regeln einer Sprachgemeinschaft sind umfangreicher und flexibler als die einer Programmiersprache. Wir können jedoch mit computerlinguistischen Methoden Elemente und Strukturen in natürlichsprachlichem Text ermitteln und so Funktionen spezifizieren, die auf diesen Einheiten operieren. Solche Funktionen können entsprechend der in Abschnitt 4.3 vorgeschlagenen Klassifizierung spezifiziert werden.

Autoren redigieren geschriebenen Text, bevor der aktuell zu schreibende Satz beendet ist. Tiefe syntaktische Analyse für die Erkennung und Kategorisierung linguistischer Einheiten in einem Text während des Schreibens ist darum nicht möglich. Wir beschränken uns daher auf möglichst flache Analysen. In Kapitel 6 werden wir entsprechende Ressourcen auswählen. Zunächst werden wir uns in Kapitel 5 noch einmal mit der Schreibforschung beschäftigen. Die Entscheidung, welche der prinzipiell denkbaren Funktionen tatsächlich implementiert werden sollten, wollen wir möglichst auf empirische Erkenntnisse stützen. In Kapitel 7 zeigen wir dann konkrete Implementierungen entsprechender Funktionen.

5 Einfluss des Schreibwerkzeugs auf den Schreibprozess

> *Yet to improve a text, writers must succesively make a series of corrections, while checking to see that each one is compatible with the others, often located at different linguistic levels.*
>
> —Annie Piolat, *Effects of Word Processing on Text Revision*, 1991

> *The use of computers to support writing is in its infancy, both in terms of the technology and our understanding of how writers write.*
>
> —Patrik O'Brian Holt, *Computers and Writing. State of the Art*, 1992

In dieser Arbeit untersuchen wir, wie die Implementierung von Funktionen, die auf den strukturellen Einheiten einer natürlichen Sprache operieren, dazu beitragen können, typische Redigierfehler beim Bearbeiten von natürlichsprachlichen Texten in Textbearbeitungsprogrammen zu vermeiden. In Anhang A haben wir entsprechende Fehler in deutschen Texten zusammengetragen, die als *slips* klassifiziert werden können.

Wir beschäftigen uns mit der *Vermeidung* solcher Fehler und haben dafür in Kapitel 4 unseren Lösungsansatz der linguistisch basierten Redigierfunktionen vorgeschlagen. Für die konkrete Implementierung solcher Funktionen ist es notwendig, die Entstehung von Fehlern wie in Anhang A zu ermitteln. Nur für einige dieser Fehler können wir im Nachhinein die Ursachen bestimmen – weil die Redigieraktionen von den jeweiligen Autoren berichtet wurden oder

in Protokollen der verwendeten Editoren nachverfolgt werden können. Wir sind jedoch an systematischeren empirischen Aussagen interessiert, die sich für die Konzeption von Editierfunktionen verwenden lassen.

Wie in Abschnitt 1.2 bereits angesprochen, wollen wir daher die Erkenntnisse der Schreibforschung verwenden, um zu entscheiden, welche Redigieraktionen durch spezifische Editorfunktionen unterstützt werden können. Dazu werden wir in diesem Kapitel auf die Beziehung zwischen Schreibprozess und Schreibwerkzeug genauer eingehen und erörtern, welche der Erkenntnisse der Schreibforschung für unsere Zwecke verwendet werden können.

Bevor wir in Abschnitt 5.3 den Einfluss des Schreibwerkzeugs auf den Schreibprozess genauer untersuchen, beschäftigen wir uns in Abschnitt 5.2 mit Textbearbeitungsprogrammen, also dem elektronischen Schreibwerkzeug selbst. Im folgenden Abschnitt 5.1 wenden wir uns zunächst der Rolle des Schreibwerkzeugs innerhalb des Schreibprozesses zu.

5.1 Die Rolle des Schreibwerkzeugs

Mit Blick auf den entstehenden Text stellt Müller [2004] in der Zusammenfassung einer Studie mit Studenten zwischen 1990 und 2000 im Fachbereich Erziehungswissenschaften der Universität Hamburg fest, «[…] dass das Schreiben mit Textverarbeitungsprogrammen per se nicht zu Verbesserungen führt, sondern Verstöße gegen Ausdruck, Stil und Grammatik sogar provoziert.» [Müller 2004, S. 156] Eine eher kritische Sicht des Einflusses des Schreibwerkzeugs. Interessanterweise wird hier aber eher das Schreib*produkt* als der Schreib*prozess* bewertet.

Die Vielfalt der Funktionen in Textbearbeitungsprogrammen und vor allem die Möglichkeiten, jederzeit an eine bestimmte Textstelle zu gelangen und dort das Schreiben fortzusetzen oder etwas zu ändern, unterstützt sehr unterschiedliche Schreibstrategien. In Befragungen äussern professionelle Autoren (sowohl aus dem Bereich des fiktionalen als auch des nichtfiktionalen und akademischen Schreibens), dass der Computer den Schreibstil nicht ändere, die Arbeit damit aber mehr Spass mache und Zeit spare, wie etwa Dorner [1992] zeigt. Zu Beginn der 1980er Jahre kommt auch das Gefühl hinzu, ein völlig neues Medium zu benutzen und ein neues Werkzeug zu beherrschen. Zu dieser Zeit ist das Werkzeug tatsächlich neu, zu Beginn des 21. Jahrhunderts jedoch nicht mehr. Heute muten Ausführungen wie von Daiute seltsam und eher anekdotenhaft an:

> Computer users tend to feel in control of the writing process. They are instructing a powerful machine to carry out massive and tedious editing tasks instantly. These tasks would require enormous amounts of the writer's engergy. For example, a student writing a story decided to change the name of the main character from Sara to Rachel. When she learned she could change all one hundred occurrences of the name with one replacement command, avoiding recopying, she was so excited that she continued her writing session for several hours beyond what she had originally planned. [Daiute 1983, S. 143]

Collier und Werier [1995] untersuchten Mitte der 1990er Jahre «[…] if the prolonged use of a word-processing system might create dependencies in experienced writers.» [Collier und Werier 1995, S. 48]. Sie konnten beobachten, dass sich die Schreibstrategien der Versuchspersonen – die im Alltag jeweils ausschliesslich mit dem Computer schrieben – stark voneinander unterschieden. Jedoch war für jede einzelne Person kein wesentlicher Unterschied festzustellen, ob sie nun mit Papier und Stift oder mit einem Textbearbeitungsprogramm schreiben musste. Das Schreiben auf Papier erschwert offenbar Schreibstrategien, die darauf beruhen, den Text exzessiv umzuschreiben, also ein Wachsen «von innen heraus» – eine grobe Gliederung wird angereichert zu Kapiteln, diese werden in einem weiteren Durchgang erweitert zu mehreren Absätzen etc. Ein Textbearbeitungsprogramm stellt sowohl dem linearen Schreiben, dem Schreiben mit häufigen Revisionen als auch dem anreichernden Schreiben geeignete Mittel zur Verfügung. [Vgl. Collier und Werier 1995]

Ein neues Schreibwerkzeug erfordert natürlich auch andere und neue Fertigkeiten von den Autoren. Der Umgang mit der Tastatur, konkret die (Un)Fähigkeit, mit zehn Fingern zu tippen, hat einen wesentlichen Einfluss auf den Schreibprozess:

> Effects attributed to this problem include conscious selection of short words and simple sentence structures, a general tendency toward summary in preference to a fuller exposition of ideas and (c) [sic!] conscious rejection of certain character combinations found to be awkward, particularly those involving the outer two fingers. [Dowling 1994, S. 229]

Der Umgang mit der Computertastatur unterscheidet sich nicht sehr vom Umgang mit der Tastatur von Schreibmaschinen, wenn auch heutige Tastaturen wesentlich weniger physische Beanspruchung beim Anschlag verursachen – wir finden also tatsächlich einen positiven physischen Effekt, wie ihn Eisenberg [1992, S. 280] vorhersagt. Nicht nur die Fertigkeit in der Bedienung der Tastatur, sondern auch der Umgang mit der Maus beeinflussen das Schreiben: «[…] the driving of the mouse itself was identified by several of the writers as providing an awkward distraction from the writing task, and, hence, as contributing to the overall difficulty of the enterprise.» [Dowling 1994, S. 230] Hinzu kommt, dass ein erzwungener ständiger Wechsel von der Tastatur zur Maus und umgekehrt die Bedienung von Textbearbeitungsprogrammen verlangsamt, die Effizienz also sinkt [vgl. Raskin 2000].

Ist es für einen Autor nicht notwendig, beim Schreiben auf die Tastatur zu sehen – kann er also «blind» schreiben (engl. als *touch typist* oder *monitor gazer* bezeichnet) –, sind die Pausen zwischen einzelnen Tastendrücken kurz. Es können mehr Wörter pro Minute geschrieben werden und es wird weniger Zeit für eine Schreibaufgabe benötigt, obwohl mehr Text produziert wird als von Autoren, die jeweils auf die Tastatur sehen müssen (engl. *keyboard gazers* oder *hunt-and-peck*-Schreiber). Die Eingabe von Text ist gewissermassen automatisiert und weniger kognitive Kapazität wird für diese Aufgabe benötigt [Johansson et al. 2010]. Ein hoher Wert für «Anschläge pro Minute» garantiert jedoch noch nicht eine geringe Fehlerquote oder einen optimalen Eingabeprozess, wie Grabowski [2008] berichtet. Erst wenn tatsächlich Tastaturschreiben erlernt und gefestigt wurde, ist es möglich, den Eingabeprozess zu automati-

sieren, die Tastatur effizient zu bedienen, weniger Fehler zu produzieren und tatsächlich «blind» zu schreiben [Grabowski 2008, S. 50].

> [The fact, that – C.M.] monitor gazers perform significantly more backspaces than keyboard gazers while reading also suggests that they frequently edit in parallel with reading. Moreover, this also indicates that monitor gazers can view and edit their text immediatly, while keyboard gazers have no other choice but to shift their attention from the keyboard to the monitor in order to read their texts. [Johansson et al. 2010, S. 848]

Wenn wir davon ausgehen, dass Autoren sich für geeignete und ergonomische Eingabegeräte – also Tastatur und Maus – entscheiden und im Umgang mit ihnen geübt sind, sodass eine effiziente Bedienung vorausgesetzt werden kann, dann bleibt noch die Frage offen, welche Bedingungen ein Textbearbeitungsprogramm erfüllen muss, um den Schreibprozess optimal zu unterstützen. Im folgenden Abschnitt 5.2 gehen wir darauf genauer ein. In Abschnitt 5.3 betrachten wir die aktuellen Erkenntnisse der Schreibforschung hinsichtlich des Einflusses von Textbearbeitungsprogrammen – also des Schreibwerkzeugs – auf den Schreibprozess.

5.2 Anforderungen an Textbearbeitungsprogramme

Taylor [1987] stellt erstmals explizit Anforderungen an Schreibprogramme, die er aus Beobachtungen von Schreibübungen in Computerräumen ableitet. Auch wenn die beobachteten Personen Studenten waren und die Mitte der 1980er Jahre zur Verfügung stehenden Schreibprogramme sich in ihrer Funktionalität und in ihrem Erscheinungsbild stark von den heutigen unterscheiden, sind diese Anforderungen immer noch aktuell.

Zum einen soll das ideale Textbearbeitungsprogramm den Autor im Hintergrund unterstützen und sich nicht aufdrängen [Taylor 1987, S. 80f]. Zahlreiche Untersuchungen zum Schreibprozess stützen diese Forderung. Immer mehr Menü-Einträge, sich öffnende Fenster, neue Funktionen, die miteinander interagieren, etc. benötigen immer mehr Speicherkapazität und immer schnellere Prozessoren – stellen also wachsende Anforderungen an die Hardware. Für den Benutzer wird die Bedienung solcher Programme ebenfalls immer anspruchsvoller. Wie in Abschnitt 2.2 schon gezeigt, sind die menschlichen kognitiven Ressourcen jedoch begrenzt und können – im Gegensatz zum Speicherplatz in Computern – nicht vergrössert werden.

Zum anderen dürfen die Funktionen des Textbearbeitungsprogramms und deren Bedienung den Schreibprozess nicht beeinträchtigen [Taylor 1987, S. 80f]. Diese Anforderung wird von aktuellen Programmen eher nicht erfüllt, auch wenn sie dieses Ziel verfolgen. Ein bekanntes Beispiel ist hier *MS Office*:

> The AI approach was aimed at fully supporting users' existing ways of working, requiring no change on their part. Its influence can still be seen in the continued development of so-called intuitive interfaces, those that anticipate a user's desired actions and

> either act directly or present a small list of options. Such interfaces, though, often end up frustrating users when the system has incorrectly anticipated a user's wishes or is seen as patronizing (Microsoft Word's "Clippy" agent being the most glaring example of the latter). [Van Ittersum 2008, S. 156f][1]

Selbst professionelle Autoren, die intensiv mit Textbearbeitungsprogrammen arbeiten, wissen in der Regel nicht, welche Möglichkeiten diese Programme offerieren [vgl. Dorner 1992, S. 8]. Immer neue Funktionen in Textbearbeitungsprogrammen führen dazu, dass Benutzer nicht wissen, welche Funktionen wo zu finden sind und was genau sie bewirken. Der WYSIWYG-Ansatz ermöglicht einerseits, jederzeit zu sehen, wie sich das Geschriebene später (vermutlich) im Druck darstellen wird. Andererseits sind Strukturen des Textes oder Markup nicht direkt sichtbar – Änderungen im Text können diese Strukturen betreffen und so unbeabsichtigt sehr störende Seiteneffekte verursachen. WYSIWYG wird daher auch als WYSIWYMG (*What-you-see-is-what-you-might-get*) oder sogar als WYSINWYW (*What-you-see-is-not-what-you-want*) [Holmes 2001, S. 126] bezeichnet.

Die Beobachtung, dass ein Übermass an Funktionen und möglichen Konfigurationen den Benutzer beeinträchtigt statt ihn zu unterstützen, trifft generell für Computerprogramme zu und ist nicht auf Textbearbeitungsprogramme beschränkt. Textbearbeitungsprogramme sollten generellen Regeln für das Design von Bedienoberflächen folgen, wie sie Raskin formuliert:

> We must design interfaces that (1) deliberately take advantage of the human trait of habit development and (2) allow users to develop habits that smooth the flow of their work. *The ideal humane interface would reduce the interface component of a user's work to bengin habituation. Many of the problems that make products difficult and unpleasant to use are caused by human-machine design that fails to take into account the helpful and injurious properties of habit formation.* One notable example is the tendency to provide many ways of accomplishing the same task. Having multiple options can shift your locus of attention from the task to the choice of method. [Raskin 2000, S. 20][2]

Eine weitere Facette zu störenden Aspekten von Schreibprogrammen liefert Sharples [1994]. Hauptsächlich beschäftigt er sich hier mit dem Rhythmus des Schreibens, der aus Folgen von eigentlichem Schreiben und Reflektieren über das Geschriebene und noch zu Schreibende besteht. Dieser Rhythmus ist von Autor zu Autor verschieden; Aktionen, die die eine – das Schreiben – oder die andere Phase – das Reflektieren – unterbrechen oder verkürzen, werden generell als den Schreibprozess behindernd empfunden. Die Verwendung von Textbearbeitungsprogrammen spielt dabei eine grosse Rolle: Zum einen treten Störungen auf, weil verschiedene automatische Prüfprogramme – wenn als `check-as-you-type` (überprüfen während des Schreibens) konfiguriert – *während* des Schreibens Meldungen liefern. Zum anderen bieten Textbearbeitungsprogramme zahlreiche Funktionen, die weder der einen noch der anderen

1. «Clippy» heisst in der deutschen Version «Karl Klammer».
2. Hervorhebung im Original.

Phase eindeutig zuzuordnen sind, wie das Zählen von Wörtern oder das Anzeigen der Struktur. «And with a wealth of displacement activities they may well distract a writer from the business of creating text.» [Sharples und van der Geest 1996, S. VI] Diese Funktionsflut nimmt in jeweils neueren Versionen von Textbearbeitungsprogrammen immer weiter zu: «[…] the development of word processors is very much a history of incremental additions to basic text editors, combined with a desire to add new twists and wrinkles for marketing purposes.» [Williams 1990b, S. 14] Neue Funktionen sind jeweils ein Werbeargument für die Hersteller, wie es schon für frühere Textbearbeitungsprogramme galt [vgl. Bergin 2006a,b, Eisenberg 1992]. In der Regel wird nicht aus der Sicht der Benutzer, sondern mit Blick auf das technisch Machbare entwickelt.

Sharples und Pemberton [1990] halten ebenfalls fest, dass Software sich entwickelt, indem auf vorangegangene Versionen aufgebaut wird. Notwendig sei jedoch eine völlig neue Implementierung, um die Bedürfnisse von Autoren zu befriedigen: «Firstly writing software must support the writer's normal working practices, and secondly, it must provide the writer with support for "externalizing cognition".» [Sharples und Pemberton 1990, S. 37] Sie zeigen auf, welchen Anforderungen Software für Autoren genügen muss, wenn man von den Bedürfnissen der Autoren und nicht vom technisch Machbaren ausgeht, siehe [Sharples und Pemberton 1990, S. 37ff].

Schreibunterstützung soll verschiedene Modi zum Editieren, Prüfen oder Ansehen von Kommentaren aufweisen, siehe [Holt 1989, S. 57] – eine Forderung, die in den heutigen Versionen komplexer Textbearbeitungsprogramme umgesetzt scheint. Ebenso die Forderung, dass es eine ständige Überprüfung und Analyse geben sollte, die zu Kommentaren und der Hervorhebung der Fundstelle führt. Die Fundstelle kann sofort oder zu einem späteren Zeitpunkt überarbeitet werden. Oakman spricht ebenfalls die Notwendigkeit an, dass die Analyse im Hintergrund ablaufen soll. Der Autor soll sich die Ergebnisse bei Bedarf ansehen und den Text entsprechend redigieren können [Oakman 1994, S. 233]. Unterstützung fordert Holt [1989] auf den Ebenen von Wörtern (hinsichtlich Orthographie und Vokabular), Sätzen (hinsichtlich Satzanfängen und Satztypen), Absätzen (hinsichtlich der Strukturierung und Satzkonstruktion), Kapiteln und dem Text als Ganzem. Schwierigkeiten in der Umsetzung sieht er im Design der Interaktion zwischen Software und Autor. Dies erfordere ein Schreibmodell, das auf diesen Aspekt Bezug nimmt [Holt 1989, S. 58].

Der Umsetzung einzelner Aspekte widmeten sich in den 1980er und 1990er Jahren verschiedene Projekte, die wir in Abschnitt 3.2 vorgestellt haben. Zumeist gingen diese Projekte nicht von vorab spezifizierten Anforderungen aus, sondern ermittelten während oder sogar erst gegen Ende der Entwicklung, welche Unterstützung nötig wäre, siehe [Williams 1989, S. 5f], [Williams 1990b, S. 14ff] und [Sharples 1996b, S. 97 und S. 110]. Alle Autoren betonen in abschliessenden Berichten, dass der Einbezug der Zielgruppe notwendig gewesen wäre – die Spezifizierung der Funktionen und Editoren stützte sich jedoch auf Annahmen, Selbstbeobachtungen und Analysen bestehender Funktionen in verfügbaren Textbearbeitungsprogrammen.

Solche Analysen beziehen sich auf (a) technische Probleme mit Hard- oder Software allgemein (etwa Abstürze des Programms oder des Computers, verschiedene Dateiformate und damit Inkompatibilität), (b) Verständlichkeit der

Terminologie von Funktionen (für automatische Prüfprogramme[3] sei die Terminologie «zu linguistisch»), (c) Dokumentation von Software, Hilfetexte, (d) unerfüllbare Erwartungen an automatische Prüfprogramme oder Nachbearbeitungssysteme, Autoren müssten die Prinzipien von solchen Programmen kennen, um sie sinnvoll nutzen zu können. [Vgl. Williams 1990b, S. 14ff]

Vergleichen wir die Anforderungslisten, die Williams [1989] und Sharples [1996b] geben, finden wir viele Gemeinsamkeiten und können sie klassifizieren als Anforderungen, die (a) den Umgang mit Dateien (etwa Versionskontrolle), (b) die Gestaltung der Bedienoberfläche (etwa verschiedene Ansichten auf ein Dokument entsprechend unterschiedlichen Informationsbedürfnissen), (c) die Arbeit mit dem Text auf verschiedenen Ebenen entsprechend verschiedenen Phasen des Schreibprozesses, (d) das Editieren des Textes und (e) die Überprüfung der Einhaltung von Bedingungen unterschiedlichster Art (etwa hinsichtlich Orthographie oder hinsichtlich einer vorgegebenen Struktur) betreffen [Sharples 1996b, S. 110]. Die entsprechenden Unterstützungen müssten interaktiv und jederzeit sofort verfügbar sein und in übliche Textbearbeitungsprogramme integriert werden können [Williams 1989, S. 5f].

Erstaunlich an solchen Auflistungen ist nicht die Fülle von Anforderungen, sondern die Tatsache, dass sie explizit als *künftig* für Entwicklungen zu berücksichtigen gekennzeichnet sind. Die einzelnen Funktionen entsprechen jedoch sehr genau den Möglichkeiten des Systems *AUGMENT*[4] aus den frühen 1960er Jahren [Callender 1982, Engelbart 1962]. Hausdorf [2005] entwickelt in seiner Dissertation *ScientiFix* als *Textproduktionssystem*, das auf allen genannten Ebenen die jeweils aktuellen Forschungsergebnisse berücksichtigt. *ScientiFix* ist jedoch nicht mehr verfügbar. In kommerziellen Textbearbeitungsprogrammen wie MS *Word* sind entsprechende Funktionen bis heute nicht vorhanden. Man muss sich immer noch Carlson anschliessen:

> I have never used a CAW [Computer-Aided Writing – C. M.] product that, eventually, didn't pinch and constrain my writing process by being distractingly intrusive, nit-picking atomistic, or downright tedious and misleading in the advice it returned. [Carlson 1990, S. 96]

Wir können zusammenfassen, dass sowohl von Benutzern von Textbearbeitungsprogrammen als auch von Softwareentwicklern fundierte und nachvollziehbare Anforderungen an Textbearbeitungsprogramme gestellt werden. Allerdings werden diese Anforderungen von aktuellen Textbearbeitungsprogrammen nicht erfüllt, obwohl die Entwicklung im Bereich Hardware es erlauben würde, solche Anforderungen umzusetzen: «Although the improvements in computer hardware have been impressive, the evolution and proliferation of software, especially word processing systems, is nothing short of phenomenal.» [Bergin 2006a, S. 33]

Die Anforderungen wurden aus verschiedenen Studien zum Schreibprozess und zum Schreibprodukt extrahiert und in früheren Jahrzehnten wurde zudem mehrfach betont, welchen Einfluss die «neuen» Schreibwerkzeuge, also die sich gerade entwickelnden Textbearbeitungsprogramme, auf den Schreibprozess

3. Siehe Abschnitt 3.3.
4. Mehr zu *AUGMENT* und *NLS* (oN-Line System) in Abschnitt 3.1.

haben. Es bleibt die Frage offen, wie diese Situation heute eingeschätzt wird. Welche Konsequenzen ergeben sich für den Schreibprozess, für Empfehlungen zu Schreibstrategien und für die Produkte, die damit geschriebenen Texte, da das Schreibwerkzeug sich offenbar nicht in die gewünschte Richtung entwickelt?

5.3 Schreibwerkzeug und Schreibprozess in der Schreibforschung

Für den US-amerikanischen Raum stellt die Zeitschrift «Computers and Composition»[5] einen Fixpunkt dar, hier werden in dem Gebiet wichtige Studien und Erkenntnisse veröffentlicht. In frühen Ausgaben aus den 1980er Jahren liegt der Fokus auf Untersuchungen, die sich mit dem Schreibprozess aus «handwerklicher» Sicht beschäftigen. Mit dem Aufkommen von Textbearbeitungsprogrammen werden Fragen zu geeigneten Programmen, Funktionen und didaktischen Szenarien gestellt und entsprechende Studien durchgeführt. Autoren werden während des Schreibens beobachtet, um zu erfahren, welche Handlungen sie vornehmen und wie Textbearbeitungsprogramme sie dabei unterstützen (können). Es wird vermutet, dass der Wechsel des Schreibwerkzeugs den Schreibprozess und insbesondere das Redigieren beeinflusst. Dazu werden zahlreiche vergleichende Studien – Schreiben mit Textbearbeitungsprogrammen und Schreiben mit Papier und Stift – durchgeführt, Hawisher [1986, 1988] und Hawisher et al. [1995] bieten einen guten Überblick.

In den aktuellen Ausgaben von «Computers and Composition» zu Beginn des 21. Jahrhunderts wie auch in den Vorträgen und Online-Beiträgen an der US-amerikanischen Konferenz «Computers and Writing»[6] verschiebt sich der Blickwinkel hin zum Schreiben als kognitive Leistung: Der Einsatz von «Multimedia» wird wichtig, Aspekte des intellektuellen Austauschs und der Wahrnehmung von Geschriebenem. Es geht weniger um die Arbeit *am* Text als vielmehr um die Arbeit *mit* Text oder die Leistung, die zu erbringen ist, um überhaupt etwas schreiben zu können. Vielfach ist von «digital writing» die Rede. Gemeint ist dabei jedoch nicht das Schreiben mit einem Computerprogramm anstelle des Schreibens mit Papier und Stift, sondern die Verwendung verschiedener digitaler Medien, um eine Äusserung zu produzieren.

So ändert sich auch das Verständnis von *Text*: «The word *text* is inadequate when discussing documents created for and about information technology. Today's interactive, hypertextual documents—many of which reside on the Internet—use color, sound, images, video, words, and icons to express their messages.» [Geisler et al. 2001, S. 282][7] Der Rückbezug auf den Schreibprozess und auf Schreibmodelle oder eine Erörterung, wie sich deren Verständnis und Bedeutung durch den Einsatz von *Flash*, Videos, *Weblogs*, *Twitter* und ähnlichem ändern, wird nicht geleistet. Eine Hinterfragung der Möglichkeiten und Grenzen der verwendeten Werkzeuge und deren Einfluss auf den Schreibprozess erfahrener Autoren wie auf didaktische Aspekte beim Lehren des Schreibens

5. «Computers and Composition is a professional journal devoted to exploring the use of computers in composition classes, programs, and scholarly projects. It provides teachers and scholars a forum for discussing issues connected to computer use. The journal also offers information about integrating digital composing environments into writing programs on the basis of sound theoretical and pedagogical decisions and empirical evidence.» `http://computersandcomposition.osu.edu` (zuletzt besucht am 8.12.2010, 19:34).
6. Besucht im Juni 2009 an der University of California, Davis.
7. Hervorhebung im Original.

findet nicht statt. Bezüge zu älteren Aufsätzen, die in den 1980er Jahren diese Fragen stellen und im Hinblick auf zu dieser Zeit aktuelle Programme beantworten und durchaus kritisch beleuchten, werden nicht hergestellt.

Für Europa sind eher die Aktivitäten der Protagonisten der *SIG Writing*, einer Gruppe innerhalb der *European Association for Research on Learning and Instruction* (EARLI), massgebend. Ein Teil der Forscher, die heute in *SIG Writing* aktiv sind, war Ende der 1980er, Anfang der 1990er Jahre in Europa in Aktivitäten im Umfeld der Konferenz «Computers and Writing» engagiert. Diese hauptsächlich von Grossbritannien ausgehenden Aktivitäten schliefen jedoch ein – begründet durch veränderte Interessen und eine zu geringe Anzahl an interessierten Forschern.[8]

Wichtige Veröffentlichungen der europäischen Schreibforschung finden sich in den Bänden der Serie «Studies in Writing»[9] sowie neu in der Online-Zeitschrift «Journal of Writing Research»[10]. Dabei finden sich sowohl europäische Beiträge in «Computers and Composition» als auch amerikanische Beiträge in den Bänden von «Studies in Writing» oder dem «Journal of Writing Research», das durch die Verleihung des *John R. Hayes*-Awards an internationaler Bedeutung gewinnt.

Wie in den amerikanischen finden wir auch in den europäischen Arbeiten keine Beiträge, die sich tatsächlich mit dem Schreibwerkzeug auseinandersetzen. Dieser aus den veröffentlichten Arbeiten gewonnene Eindruck bestätigt sich in den Vorträgen an der biennalen Konferenz der *SIG Writing*[11]. Amerikanische Schreibforschung ist hauptsächlich durch Interessen innerhalb der universitären Schreibkurse (*composition courses*, z. B. *Writing across the curriculum* (WAC) oder *Writing in the Disciplines* (WID)) motiviert und innerhalb der Schreibzentren (*Writing Centers*) oder der Institutionen für Englische Philologie verortet. Der Fokus liegt auf pädagogischen Aspekten zum Erlernen und Verbessern des Schreibens. Europäische Schreibforschung ist dagegen hauptsächlich in psychologischen Instituten verortet, kognitive Aspekte spielen eine viel grössere Rolle.

Studien zur Analyse von Tastendrücken beschäftigen sich vorrangig mit der Entwicklung des *Textes* während des Schreibprozesses, etwa von Sullivan und Lindgren [2006] und Van Waes et al. [2006, 2009]. In Studien, in denen der Einfluss des Computers auf den Schreibprozess untersucht wird, wie von Johansson et al. [2008, 2010], Piolat et al. [2004, 1997] und Severinson Eklundh [1994], oder die mentale Prozesse betrachten, wie von Chenoweth und Hayes [2003], werden entweder keine Angaben zum eingesetzten Werkzeug gemacht oder es werden prototypische Editoren mit einem sehr eingeschränkten Funktionsumfang erwähnt.

8. Persönliche Kommunikation mit Mark Torrance, 10. September 2010.
9. http://www.ilo.uva.nl/Projecten/Gert/StudiesInWriting/Writing_book_series.htm (zuletzt besucht am 8.12.2010, 19:34)
10. «The Journal of Writing Research (JoWR) is a new international peer reviewed journal that publishes high quality theoretical, empirical, and review papers covering the broad spectrum of writing research. The mandate of the Journal of Writing Research is: to publish excellent and innovative writing research drawn from a range of academic disciplines (e.g. psychology, linguistics, pedagogy, design studies, communication studies, information and communication technology, learning and teaching); to stimulate interdisciplinary writing research; to be fully international; to apply high academic standards, including double blind peer review; to share knowledge through open access.» http://jowr.org/ (zuletzt besucht am 8.12.2010, 19:34).
11. Besucht im September 2010 in Heidelberg, Deutschland.

Es ist jedoch bekannt, dass der Schreibprozess durch verschiedenste Umstände und Aspekte beeinflusst wird und Autoren Wert auf verschiedene Dinge legen, um «gut» schreiben zu können (siehe auch [Sharples 1996a, S. 139]):

> Each [writer – C.M.] has a personal set of habits, rituals and techniques which include a variety of individual, idiosyncratic approaches. Many report that the practical "nuts and bolts" aspects of composition—what type of pencil or pen to use, bying paper and expensive cartridges for the printer, getting situated happily and comfortable at one's desk, having hot coffee nearby and J. S. Bach on the radio—are equally important parts of the process. [Calonne 2006, S. 149]

Die Kategorisierung von Strategien bezüglich des *gesamten* Schreibprozesses bezog das Schreibwerkzeug ursprünglich nicht ein. Die Unterscheidung in Schreibtypen wie *Mozart* oder *Beethoven* [Boehm 1993, Bridwell 1980] bezieht sich auf den Schreibprozess per se ohne Berücksichtigung des Schreibwerkzeugs oder des Schreibmediums. Erst Van Waes [1992] untersucht explizit Schreibstrategien bei der Verwendung von Textbearbeitungsprogrammen. Er ermittelt fünf verschiedene Schreibtypen und beobachtet Verhaltensanpassungen, die Schreiber eines bestimmten Typs mit einem anderen Schreibwerkzeug vornehmen [Van Waes 1992, S. 184f].

Exkurs: Autoren und ihre bevorzugten Schreibwerkzeuge
Dass die Wahl des Schreibwerkzeugs wichtig ist, zeigt die Antwort von Don DeLillo 2010 auf die Interviewfrage «Für Sie stellt der PC also keine Versuchung dar»:

> Nein – ich habe meine alte, mechanische Schreibmaschine, die ich schon als Occasion gekauft habe, und ohne die könnte ich schwerlich arbeiten. Ich brauche dieses konkrete Gefühl – Hände auf den Tasten und Hämmer, die auf die Seite schlagen; ich brauche Papier, keinen Bildschirm. Ich muss Korrekturen von Hand machen können, während das Papier noch in der Maschine steckt, ich muss die Walze drehen können.[12]

Dagegen arbeitet Neal Stephenson mit dem Computer und berichtet 1999 in «In the beginning was the command line» [Stephenson 1999]:

> I began using Microsoft Word as soon as the first version was released around 1985. After some initial hassles I found it to be a better tool than MacWrite, which was its only competition at the time. I wrote a lot of stuff in early versions of Word, storing it all on floppies, and transferred the contents of all my floppies to my first hard drive, which I acquired around 1987. As

12. Neue Zürcher Zeitung, 20. Februar 2010, S. 63f, «Im Anfang war das Wort. Der amerikanische Schriftsteller Don DeLillo spricht über seinen neuen Roman und das Handwerk des Schreibens», Interview: Angela Schader. (Es ist keine Information zur Sprache des Interviews und allfälliger Übersetzung vorhanden.)

> new versions of Word came out I faithfully upgraded, reasoning that as a writer it made sense for me to spend a certain amount of money on tools.
>
> Sometime in the mid-1980's I attempted to open one of my old, circa-1985 Word documents using the version of Word then current: 6.0 It didn't work. Word 6.0 did not recognize a document created by an earlier version of itself. By opening it as a text file, I was able to recover the sequences of letters that made up the text of the document. My words were still there. But the formatting had been run through a log chipper—the words I'd written were interrupted by spates of empty rectangular boxes and gibberish. [Stephenson 1999, S. 70][13]

Auch über andere Autoren gibt es Aussagen zu ihren Schreibwerkzeugen – hier der Schreibmaschine:

> Olivetti-Schreibmaschinen – nicht nur jene von Cormac McCarthy – sind legendäre Maschinen. Wolfgang Koeppen hat auf einer geschrieben, Günter Grass bunkert wohl gleich mehrere für harte Zeiten und Stephen King hat seinen ganz persönlichen Horror mit einer Olivetti erlitten, die er in jungen Jahren und lange vor seinem grossen Erfolg beim Schreiben im Heizraum eines Wohnwagens auf den Knien balancieren musste, weil er nicht genügend Platz hatte, sie irgendwo hinzustellen.
>
> «Die Maschine diszipliniert meine Arbeit und gibt dem Satz von Anfang an Klarheit», sagte Koeppen über sein Modell. Auch McCarthys Bücher wären ohne den sanften Anschlag und das leise Klackern der Typen wohl andere Werke geworden. [14]

Ebenso finden wir Äusserungen über die Verschiedenartigkeit des Schreibens mit Papier und Stift oder mit dem Computer:

> Früher, als es noch keine Computer gab, habe ich immer meine Texte zunächst mit der Hand geschrieben und sie danach in die Maschine getippt. Zwischen mir, dem Papier und meinem Füller gab es ein inniges Verhältnis. […] Ich blieb ihm immer treu, trug ihn in der Brusttasche und trennte mich nie von ihm. Selbst nachts lag er neben meinem Bett auf einem Nachttisch. […]
>
> […] die Bearbeitung eines bereits geschriebenen Textes ist durch den Computer unvergleichlich leichter geworden. Für Streichungen, Ergänzungen, Umstellungen von Textpassagen und Korrekturen benötige ich nur einen Bruchteil jener Zeit, die ich früher dafür brauchte.
>
> Natürlich haben diese Vorteile ihren Preis. Dieses innige Verhältnis, das zwischen mir, dem Pelikan-Füller

13. Vermutlich hat sich hier ein Fehler eingeschlichen und es müsste "Sometime in the mid-1990's" heissen.
14. Süddeutsche Zeitung, 2. Dezember 2009, S. 11, «All die schönen Lettern. Cormac McCarthy verkauft seine Schreibmaschine» (Johannes Boie).

> und dem Papier bestand, existiert nicht mehr. Bei aller Begeisterung für die Vorteile des Computers werde ich dieses Gefühl nicht los, etwas Vertrautes, etwas, das zu meinem Wesen, meiner Persönlichkeit gehörte, verloren zu haben.[15]

Nicholas Carr äussert sich direkt zum Redigieren am Computer:

> The computer, I began to sense, was more than just a simple tool that did what you told it to do. It was a machine that, in subtle but unmistakable ways, exerted an influence over you. The more I used it, the more it altered the way I worked. At first I had found it impossible to edit anything on-screen. I'd print out a document, mark it up with a pencil, and type the revisions back into the digital version. Then I'd print it out again and take another pass with the pencil. Sometimes I'd go through the cycle a dozen times a day. But at some point—and abruptly—my editing routine changed. I found I could no longer write or revise anything on paper. I felt lost without the Delete key, the scrollbar, the cut and paste functions, the Undo command. I *had* to do all my editing on-screen. In using the word processor, I had become something of a word processor myself. [Carr 2010, S. 13][16]

Sibylle Berg bemerkt zum Schreiben am Computer:

> Ich benutzte Computer, als sie noch so gross waren wie Fussballfelder. Unbegreiflich war mir, wie andere Berufsschreiber an Schreibmaschinen festhalten konnten, hart im Anschlag, Tipp-Ex, Bogen einspannen, rausziehen, all die Ritualsachen, die mit dem Beruf nichts zu tun hatten. Ich liebe Mails, seit ihrer Erfindung. Bedeuteten sie doch, nie mehr ängstlich um Telefone zu streichen. Ich könnte das Internet fressen vor Zuneigung, denn es heisst – kein Hocken in Bibliotheken, Fotokopieren, Suchen, keine staubigen Finger.[17]

Bekannt sind ebenfalls die Versuche Friedrich Nietzsches, wegen seiner nachlassenden Sehkraft eine Schreibmaschine zu verwenden, auf der er «blind» schreiben konnte. Sein Schreiben und auch seine Texte ändern sich, er schreibt kürzer – sowohl kürzere Sätze als auch insgesamt kürzere Texte, mehr Lyrik als Prosa. An Heinrich Köselitz schreibt er am 24. Februar 1882 (zitiert nach [Eberwein 2005, S. 122f]): «Sie haben Recht, unser Schreibzeug arbeitet mit an unseren Gedanken. Wann werde ich es über meine Finger bringen, einen langen Satz zu drucken!». Köselitz hatte ihn in einem Brief vom 19. Februar 1882 nach seinen Erfahrungen gefragt.

15. Bahman Nirumand, iranischer Schriftsteller und Publizist im Artikel «Ein Tor zur Heimat» in der Neuen Zürcher Zeitung, 18. September 2010, S. 59.
16. Hervorhebung im Original.
17. Sibylle Berg, «Schreibkurs im Netz», Neue Zürcher Zeitung, 6. November 2010, S. 59.

Abbildung 5.1: Restauriertes Patentmodell der Malling-Hansen Schreibkugel.[18]

Eberwein [2005] berichtet über die Rekonstruktion von Nietzsches Schreibmaschine – einer *Malling-Hansen Schreibkugel* (siehe Abbildung 5.1) – und der darauf entstandenen Texte. Nach einigen Wochen gibt Nietzsche die Versuche mit der Schreibmaschine auf und kehrt zum Schreiben mit der Feder zurück. Er bittet seine Schwester Anfang Juli 1882 (zitiert nach [Eberwein 2005, S. 214]):

> Diese Federn sind fürchterlich, eine wie die andere. Erweise mir die Gunst, durch Dr. Romundt ein Gros von der Humboldfeder Roeders B kommen zu lassen. Es ist die einzige Feder, mit der ich noch schreiben kann.

und wenige Tage später:

> Bitte, um Himmels Willen: Stahlfedern! Die Naumburger also: B. John Mitchells classical 689! Später mag mir Dr. Romundt die mir allein nützliche Humboldfeder B (Roeders) schaffen.

18. Abbildung mit freundlicher Genehmigung der International Rasmus Malling-Hansen Society.

> Wir finden hier also sehr genaue Anforderungen an das Schreibwerkzeug, in ihrer Explizitheit vergleichbar mit den Anforderungen, die wir in Abschnitt 5.2 für Textbearbeitungsprogramme zusammengetragen haben.
>
> **Ende des Exkurses**

Eine bewusste Wahrnehmung der Einflüsse und die sehr bewusste Auswahl der Schreibwerkzeuge wird lediglich Schriftstellern zugestanden – erfahrene und professionelle Autoren, wie etwa Akademiker oder Journalisten, sind dagegen in der Regel mit den Werkzeugen zufrieden, die ihnen zur Verfügung gestellt werden [vgl. auch Williams 1992], bzw. wird dieser Aspekt gar nicht thematisiert. Sie passen sich an das Schreibwerkzeug an und bestehen nicht auf der Anpassung des Schreibwerkzeugs an ihre Bedürfnisse. Oft sind sie sich nicht einmal dessen bewusst, dass sie ähnlich sorgfältig, wie sie die Marke und das Modell ihres Laptops auswählen, auch das Textbearbeitungsprogramm auswählen könnten. Sie erinnern sich jedoch häufig an Funktionen in älteren, heute nicht mehr erhältlichen Textbearbeitungsprogrammen, die elaboriertere Funktionen hatten, die sie heute vermissen.[19]

Wir finden in aktuellen Arbeiten der Schreibforschung keine Antworten auf Fragen, wie sie Rosson [1983] für Texteditoren untersucht:

> What are the effects of experience on text editing behavior? Do users inevitably develop optimal strategies for getting their work done? The answer to such questions are becoming increasingly important, as more and more individuals begin to use word processing equipement routinely. In the best of all possible worlds, experienced users do become experts, able to quickly and accurately choose and execute optimal procedures to accomplish any given goal. Such a state of affairs would make designers of editing systems very happy indeed. But another alternative exists, that at least some proportion of experienced users stabilize at some nonoptimal level of skill. [Rosson 1983, S. 171]

Diese Fragen lassen sich ohne Abstriche auf den Umgang mit Textbearbeitungsprogrammen übertragen. Mit den Studien von Whiteside et al. [1982] und Rosson [1983] liegen Antworten für Editoren Anfang der 1980er Jahre vor. Aussagen über Textbearbeitungsprogramme im 21. Jahrhundert gibt es jedoch nicht, vergleichbare Studien werden aktuell nicht durchgeführt. Für die Entwicklung von Texteditoren oder Programmierumgebungen finden wir jedoch aktuelle Studien, etwa von Ko et al. [2005], die untersuchen, welche Funktionen Programmierer verwenden, um bestimmte Editieraufgaben zu lösen.

Es ist unklar, inwieweit die Anforderungen an Textbearbeitungsprogramme, wie sie oben in Abschnitt 5.2 aufgelistet sind, durch die Schreiberfahrungen mit Papier und Stift – oder der Schreibmaschine – beeinflusst sind. Stellen Autoren, die fast ausschliesslich elektronische Texte verfassen, andere Anforderungen? Es ist zu beobachten, dass Autoren, die das Arbeiten mit der Schreibmaschine gewohnt sind, dazu neigen, den zu schreibenden Text vor sich hin zu sprechen

19. Interview mit Charles Bazerman, 10. September 2010.

und so verschiedene Varianten auszuprobieren, um dann eine Variante zu schreiben, die nur noch minimal redigiert wird. Dagegen äussern Autoren, die ausschliesslich mit einem Textbearbeitungsprogramm arbeiten, dass sie verschiedene Varianten eines Satzes oder Textabschnitts direkt schreiben und am Bildschirm sehen müssten, um sich für eine zu entscheiden. Hier sind viel mehr Redigieroperationen im Text selbst zu beobachten.

Für unsere Absichten ist es notwendig, die Redigieraktionen hinsichtlich ihrer Intention zu klassifizieren. Werden Operationen nur aufgrund der zu beobachtenden Veränderungen der Textoberfläche klassifiziert, kann dies dazu führen, dass strukturell identische Operationen beim jeweiligen Auftreten im Schreibprozess unterschiedlich kategorisiert werden. Auch die Beobachtung der Tastendrücke, Menüaufrufe oder Mausbewegungen allein führt dazu, dass intentionell vergleichbare Operationen unterschiedlich klassifiziert werden. Ein Autor kann das Vertauschen der Elemente einer Koordination vornehmen mit Hilfe von `cut` und `paste`, durch komplettes Löschen und Neuschreiben, eine Kombination aus beidem etc. Für uns ist jedoch hauptsächlich interessant, *dass* er die Elemente einer Koordination umordnet, um zu verstehen, wie oft und in welchen Phasen des Schreibprozesses diese Aktion durchgeführt wird, um dem Autor *eine* geeignete Funktion für die Umsetzung dieses Zieles anbieten zu können.

Gerade im europäischen Raum finden wir sehr viele Studien, die das Schreiben in kontrollierten Laborsituationen untersuchen, etwa von Grabowski [2008], Johansson et al. [2010] undWengelin et al. [2010]. Ähnliche Fragen sind bereits in früheren Studien in Vergleichen von «Schreiben mit Papier und Stift» und «Schreiben mit dem Computer» gestellt und beantwortet worden. Heute sollten diese Fragen jedoch im Hinblick auf *verschiedene* elektronische Schreibwerkzeuge – etwa Online-Editoren in Blogs oder Learning-Management-Systemen oder die aktuelle Version von *MS Office* – sowie deren Möglichkeiten und Grenzen gestellt werden. Tatsächlich finden wir Studien hinsichtlich wirklich *neuer* Schreibwerkzeuge wie Diktierprogramme, etwa von Leijten et al. [2010a], Leijten und Van Waes [2005b] und Leijten et al. [2010b]. Das Schreiben mit einem Textbearbeitungsprogramm mittels Tastatur und Maus scheint dagegen nicht mehr interessant zu sein und als gelöstes Problem betrachtet zu werden.

Die Kontrolle bestimmter Faktoren in Laborsituationen ist sicherlich notwendig, um Einflüsse auf die interessierenden Aspekte auszuschliessen. Mit Vorsicht sind jedoch Ergebnisse zu interpretieren, die aus Studien stammen, die etwa die Rolle von orthographischen Schwierigkeiten des Autors bei Tippfehlern untersuchen und den Versuchspersonen – im Gegensatz zu ihrer gewohnten Schreibumgebung – den Einsatz einer Rechtschreibkontrolle versagen, jedoch diesen Aspekt in der Auswertung der gewonnenen Daten nicht einbeziehen, wie es etwa Quinlan et al. [2009] und Leijten et al. [2010b] tun. Die Mehrheit der erwachsenen erfahrenen Autoren verwendet die automatische Rechtschreibkorrektur – dies hat natürlich einen Einfluss auf die Konzentration auf orthographisch korrektes Schreiben und den Text, der in einem ersten Schreibdurchgang produziert wird.[20] So berichtet Weder [2010, S. 92], dass ihre Probanden sich im Alltag auf die Rechtschreibkorrektur verliessen, in ihrem

20. In einem eingeladenen Vortrag an der Tagung der Gesellschaft für Medien in der Wissenschaft 2006 verwies Prof. Gráinne Conole auf den Einfluss des Computers auf Aspekte des täglichen Lebens und Arbeitens: «I don't care about spelling» – die Rechtschreibkorrektur übernimmt diese Aufgabe, kognitive Ressourcen können für etwas anderes verwendet werden.

Experiment stand eine solche jedoch nicht zur Verfügung. Beobachtet wird also Verhalten, das künstlich erzwungen ist – daraus werden dann allgemeine Schlussfolgerungen abgeleitet. Die gleiche Kritik trifft Studien zu Pausen zwischen Tastendrücken – auch hier werden die Studien mit Editoren mit sehr eingeschränktem Funktionsumfang durchgeführt, die Versuchspersonen müssen mit einer ungewohnten Tastatur arbeiten, siehe [Johansson et al. 2008] und [Wengelin et al. 2009].

Ebenso fragwürdig sind Studien, die die Tatsache nachweisen, dass das Schreibwerkzeug keinen Einfluss auf die *Qualität* des entstandenen Textes hat und Autoren ihre Schreibstrategie unabhängig vom Werkzeug verfolgen – solche Studien liegen bereits von Collier und Werier [1995] aus den 1990er Jahren vor. Neu ist hier lediglich die Möglichkeit, das Schreiben mit dem Stift elektronisch zu erfassen, indem entsprechende Software eingesetzt wird, etwa *Eye and Pen* [Alamargot et al. 2006]. Wie schon in Studien in den 1980er und 1990er Jahren werden zum Vergleich keine aktuellen Textbearbeitungsprogramme eingesetzt, sondern Editoren, die lediglich einen beschränkten Funktionsumfang besitzen, wie etwa von Johansson et al. [2010] und Van Waes et al. [2010], obwohl den Versuchspersonen gesagt wird, sie könnten alle «gewohnten» Editierfunktionen verwenden.

Wengelin et al. [2010] stellen explizite Fragen zum Aspekt des Lesens während des Schreibens: «*what* they look at, *when*, *why*, in *what contexts*, at *what stages of the writing process* […]» [Wengelin et al. 2010, S. 737][21] und beziehen sich dabei ausdrücklich auf den Text. Ähnlich präzise Fragen werden für den Umgang mit einem Textbearbeitungsprogramm bzw. hinsichtlich dessen Bedienoberfläche nicht gestellt und nicht untersucht. Dabei sind Fragen, wie «What does a writer look at any given moment? For how long? At what stage of the writing process? And finally, how does this visual behaviour relate to other parts of the writing process and in what ways does it affect the characteristics of the final text?» [Wengelin et al. 2010, S. 738], eindeutig mit dem Schreibwerkzeug verknüpft.

Die Tatsache, dass solche Studien, die die tatsächliche Benutzung von Textbearbeitungsprogrammen untersuchen, kaum vorliegen, ist umso erstaunlicher, als einer der einflussreichsten Schreibforscher – John R. Hayes – bereits 2001 festhält:

> I believe that when writing researchers become actively involved in helping to solve practical communication problems, whether in schools or government or industry, they not only bring new ideas to the task but they also discover new ideas that can take away. I believe, therefore, that a major goal of researchers in this field should be to apply what is learned through research to improve the quality of writing by **studying writing in practical settings**, by working to **improve writing pedagogy** and by **improving the tools that facilitate writing** (e.g., writing software). [Hayes 2001, S. 236][22]

Studien, in denen *keystroke-logging* verwendet wird, wie die von Leijten et al. [2010a], Perrin [2006a,b] oder Van Waes et al. [2009], benutzen Daten aus

21. Hervorhebung im Original.
22. Hervorhebungen hinzugefügt.

natürlichen Schreibsituationen, was die erste Forderung («studying writing in practical settings») erfüllt. Bereits van der Geest [1996] macht Vorschläge zu Studien von natürlichen Schreibsituationen. Und auch Severinson Eklundh und Kollberg [1996] argumentieren für weiterreichende Analysen von Aufzeichnung von Tastendrücken, um beispielsweise festzustellen, ob die sofortige Korrektur eines getippten Buchstabens tatsächlich ein erkannter Tippfehler oder die Revision der ursprünglich geplanten Fortsetzung des gerade zu schreibenden Satzes ist [Severinson Eklundh und Kollberg 1996, S. 172f].

Im Bereich der zweiten Forderung («improve writing pedagogy») finden wir Aktivitäten, wie etwa von Buck [2008]. Van Waes [1992] sowie Van Waes und Schellens [2003] widmen sich explizit dem Textbearbeitungsprogramm, um Profile von Schreibstrategien zu ermitteln, auch hier geht es jedoch nicht um die Verbesserung des Schreibwerkzeugs, wie von Hayes angeregt. Diese Forderung nach dem Einbezug des Schreibwerkzeugs («improving the tools that facilitate writing») wird von der Schreibforschung nicht aufgegriffen, obwohl bereits Holt [1992] festhält:

> It seems clear however that in order to produce computer based tools to support writers and the writing process we must increase our knowledge of how writers conduct their craft. An increased understanding of writer's requirements and the tasks involved in writing will form the basis of the next generation of writing tools. [Holt 1992, S. X]

Die Hersteller von Textbearbeitungsprogrammen widmen sich der Verbesserung dieser Programme und sind daher interessiert an Daten zur Bedienfreundlichkeit ihrer Software. Beispielsweise kann man sich als Benutzer von *MS Office* Produkten damit einverstanden erklären, dass die Verwendung von Funktionen oder das Auftreten von Fehlern registriert und an *Microsoft* gesandt wird, siehe Abbildung 5.2 auf der nächsten Seite.[23]

Solche Daten liefern Hinweise darauf, welche Funktionen wie häufig benutzt werden. Fehler sind in diesem Zusammenhang jedoch nur als Fehler der Software zu verstehen, d. h., das Ausführen einer Funktion schlägt fehl, und sind mit der Ausgabe einer Fehlermeldung verbunden. Fehler wie in Anhang A im bearbeiteten Text werden so nicht berücksichtigt. Ebenfalls wird hier keine Verbindung zwischen der Benutzung von bestimmten Funktionen, dem geschriebenen oder zu schreibenden Text und der Absicht des Autors ermittelt. Solche Daten sind also für die von uns gesuchten Informationen ebenfalls nicht verwendbar.

5.4 Zusammenfassung, Schlussfolgerungen

In diesem Kapitel haben wir den aktuellen Stand der Schreibforschung erörtert. Frühere Textbearbeitungsprogramme verfügten über Funktionen, die aktuelle Programme nicht mehr anbieten, die jedoch in Anforderungen an solche Programme sowohl von Softwareentwicklern als auch von Autoren genannt

23. Für mehr Informationen siehe `http://www.microsoft.com/products/ceip/EN-US/default.mspx` (zuletzt besucht am 8.12.2010, 19:34).

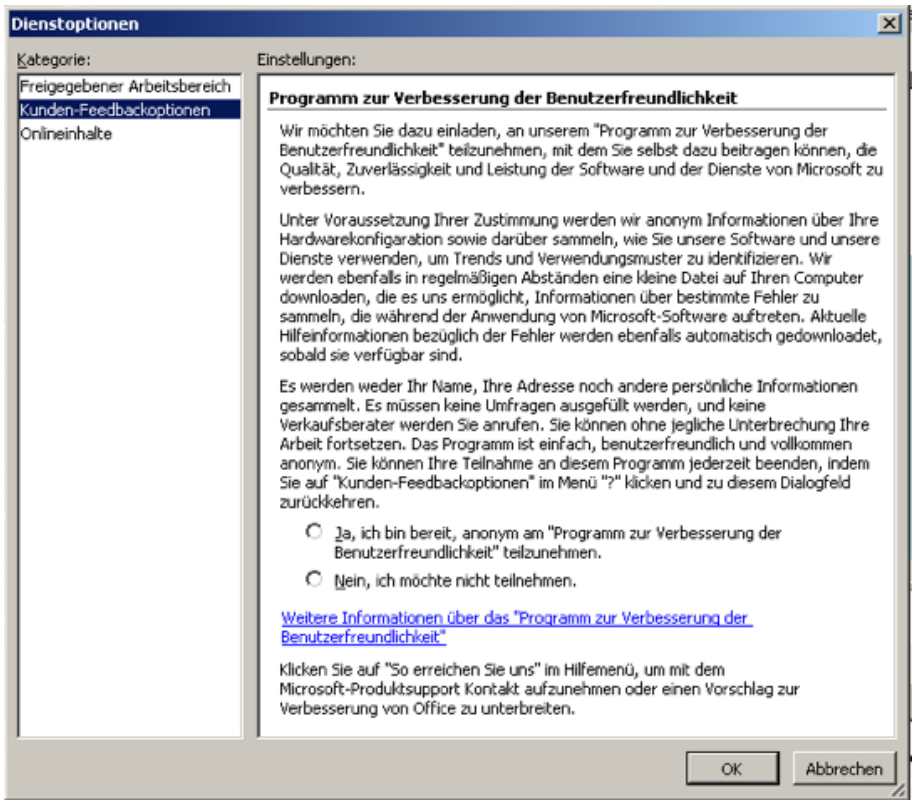


Abbildung 5.2: MS Office Programm zur Verbesserung der Bedienfreundlichkeit.

werden. Ausgehend von Äusserungen von Autoren und von Beobachtungen von Schreibprozessen ist zu vermuten, dass das Schreibwerkzeug, dessen Bedienungsmöglichkeiten und seine Funktionen den Schreibprozess und letztlich das Schreibprodukt beeinflussen. Die Schreibforschung kann dazu jedoch keine empirischen Daten liefern, da die Beobachtung des Schreibprozesses sich auf Veränderungen des *Produktes* im Zeitraum seiner Entstehung beschränkt und das Schreibwerkzeug nicht berücksichtigt.

Heutige technische Möglichkeiten erlauben es jedoch, sowohl die Entstehung des Textes als auch die Verwendung des Schreibwerkzeugs aufzuzeichnen. Aktuelle Projekte zur Auswertung solcher Daten unter Berücksichtigung beider Aspekte beginnen jedoch erst. Hier liegt eine grosse Chance der Schreibforschung, Antworten auf Fragen zu finden, die seit mehr als 15 Jahren angesprochen werden und deren Beantwortung erst eine Weiterentwicklung und eine bewusste Verwendung des Computers als Schreibwerkzeug ermöglicht.

Wir können im Moment also noch nicht Erkenntnisse der Schreibforschung verwenden, um zu bestimmen, welche Funktionen in Textbearbeitungsprogrammen notwendig sind, um *slips* wie in Anhang A zu vermeiden. Es gibt bislang keine Erkenntnisse, ob Autoren für eine bestimmte Redigieroperation jeweils eine bestimmte Sequenz von zeichenbasierten Kernfunktionen in Editoren verwenden oder ad hoc eine Lösung suchen. Für die Entscheidung, welche der in Kapitel 4 prinzipiell vorgeschlagenen Funktionen tatsächlich implementiert werden sollten, müssen wir uns auf Selbstbeobachtung stützen, ähnlich vielen der in Abschnitt 3.2 vorgestellten Projekte.

Die Erkenntnisse der Schreibforschung, dass Redigieren zu jedem Zeitpunkt innerhalb des Schreibprozesses stattfindet und Autoren bereits redigieren, bevor der aktuelle Satz vollständig geschrieben ist, sind dagegen wertvolle Informationen. Sie stützen die Designprinzipien, die wir in Abschnitt 4.4.1 dargestellt

haben: Die Funktionen müssen interaktiv bedienbar sein und zu jedem beliebigen Zeitpunkt aufgerufen werden können. Die Funktionen verwenden computerlinguistische Ressourcen, können jedoch nicht auf tiefer syntaktischer Analyse beruhen, da eben auch unvollständige Sätze bereits verändert werden.

6 Auswahl computerlinguistischer Methoden und Ressourcen

By the time I left Grenoble to come back to PARC in 2001, Inxight, a Xerox spinoff company in California, was marketing finite-state morphological analyzers and stemmers for about three dozen languages. From a computational point of view morphology was a solved problem.

—Lauri Karttunen, *Word Play*, 2007

¡Qué puedo decir! ¡qué voy a decir!
sino tengo palabras, tengo que mentir

—Ska-P, *Qué puedo decir*, 2008

In Abschnitt 1.2 haben wir festgehalten, dass ein relevanter Aspekt für den möglichen Erfolg unseres Ansatzes die Qualität der verfügbaren computerlinguistischen Ressourcen ist. Wir vermuteten, dass hier durch intensive Forschung in den letzten Jahrzehnten ein guter Stand erreicht worden ist. In Kapitel 4 haben wir das Konzept des linguistisch unterstützten Redigierens entwickelt und aufgezeigt, welche computerlinguistischen Ressourcen wir verwenden: interaktive Systeme zur automatischen Analyse und Generierung von Wortformen und flache syntaktische Analyse.

Johnson und Guilfoyle [1989] stützen sich in ihrer Vorhersage über die Entwicklung im Bereich Textbearbeitung auf die Entwicklung der Computerlinguistik

(hier als *NLP* bezeichnet): «In the longer term the total potential for NLP applications is as great as it ever was, the future has been postponed rather than cancelled.»[Johnson und Guilfoyle 1989, S. 7] Oakman [1994] fordert mehr Forschung im Bereich syntaktischer und semantischer Analyse, um automatische Prüfprogramme zu verbessern, wie wir sie in Abschnitt 3.3 betrachtet haben.

Wir untersuchen hier den aktuellen Stand der Forschung computerlinguistischer Ressourcen für das Deutsche. Dabei geht es uns nicht um eine Bewertung verschiedener theoretischer Ansätze und experimenteller Systeme zur Behandlung von exemplarischen isolierten Problemstellungen. Wir sind vielmehr an praxisrelevanten Systemen interessiert, die für unsere Zwecke im Projekt *LingURed* interaktiv eingesetzt werden können und nicht hinsichtlich Domäne, Genre oder Benutzergruppe eingeschränkt sind.

Wir bewerten daher Geschwindigkeit – für einen interaktiven Einsatz essenziell – sowie Abdeckung und Qualität der Ergebnisse, um die Verlässlichkeit der Ergebnisse für den Benutzer einschätzen zu können. Weitere Kriterien sind die Verfügbarkeit der jeweiligen Systeme und Schnittstellen, um sie mit anderen Anwendungen kombinieren zu können. Wenn möglich, wollen wir Komponenten verwenden, die frei zugänglich sind, da die resultierenden Funktionen ebenfalls frei zugänglich gemacht werden sollen.

Zunächst spezifizieren wir in Abschnitt 6.1 die Anforderungen. Wir betrachten in Abschnitt 6.2 Systeme für morphologische Analyse und Generierung, in Abschnitt 6.3 Systeme zur automatischen Wortartbestimmung und in Abschnitt 6.4 Systeme zur flachen syntaktischen Analyse. Wir stellen jeweils dar, nach welchen Gesichtspunkten die Ressourcen ausgewählt wurden, die für die in Abschnitt 7.3 dargestellten Funktionen eingesetzt werden. Dieses Kapitel zeigt einen Ausschnitt des aktuellen Stands der Entwicklung von computerlinguistischen Methoden und Ressourcen für Deutsch und deren Eignung für die Einbindung in interaktive praxisrelevante Funktionen.

6.1 Anforderungen

Die folgenden Anforderungen müssen computerlinguistische Systeme erfüllen, um als Ressource für linguistisch motivierte Editierfunktionen im *XEmacs* verwendet werden zu können.[1] Die aufgeführten Anforderungen beziehen sich spezifisch auf das Projekt *LingURed*. Einige der Anforderungen lassen sich abstrakt als generelle Voraussetzungen für die Integration von Ressourcen in bereits bestehende Programme zur interaktiven Nutzung verstehen.

Verfügbarkeit Als Editor verwenden wir *XEmacs*, der als Open-Source-Software verfügbar ist.[2] Unser Ziel ist es, alle implementierten Funktionen ebenfalls frei zur Verfügung zu stellen, sodass jeder sie nach Wunsch in seine Instanz des *XEmacs* integrieren kann. Dies bedingt, dass alle verwendeten Ressourcen ebenfalls frei verfügbar sind. «Frei verfügbar» bezieht sich auf bestimmte Lizenzformen, wie beispielsweise *GPL* oder die *Apache*-Lizenz. Es kann notwendig sein, dass ein Benutzer sich für die Verwendung einer

1. Wir gehen in Abschnitt 7.1 genauer auf *XEmacs* ein.
2. http://www.xemacs.org/ (zuletzt besucht am 8.12.2010, 19:34).

Ressource registrieren muss. Die Lizenz sollte in jedem Fall so gestaltet sein, dass darauf aufbauende Funktionen oder Applikationen an andere Benutzer weitergegeben werden dürfen.

Diese Anforderungen gelten unter idealen Bedingungen: Sollten sich zwei Ressourcen lediglich hinsichtlich der Verfügbarkeit unterscheiden, würden wir diejenige mit der offensten Lizenz wählen. Die Verfügbarkeit kann jedoch kein allein entscheidendes Kriterium sein – die von uns implementierten Funktionen sollen Autoren im Schreibprozess unterstützen, Qualität und Abdeckung einer Ressource haben daher einen höheren Stellenwert.

Die Funktionen werden so implementiert, dass es möglich ist, eine spezifische Ressource gegen eine andere auszutauschen – etwa aus Lizenzgründen oder weil zu einem späteren Zeitpunkt qualitativ hochwertigere Ressourcen zur Verfügung stehen.

Installation Die Ressourcen müssen sich relativ einfach kompilieren und installieren lassen. Die von uns implementierten Funktionen sollen nicht über einen Web-Service angeboten werden – man könnte sie dann ohne Internetzugang nicht verwenden und wäre auf die Verfügbarkeit des Web-Service, d. h. von einer externen Instanz, abhängig. Die Ressourcen sollen daher jeweils lokal auf dem Computer des Autors installiert sein. Die computerlinguistischen Systeme sollen daher auf handelsüblichen Desktop-Computern oder Laptops möglichst unabhängig vom Betriebssystem installiert werden können. Der benötigte Aufwand hinsichtlich Zeit und Rechenleistung sollte möglichst klein sein.

Schnittstellen und Format der Resultate Die computerlinguistischen Ressourcen verwenden als Eingabe entweder eine explizite Benutzereingabe oder bereits geschriebenen Text – den gesamten bereits geschriebenen Text oder einen über Parameter bestimmten Textausschnitt (z. B. den Satz, in dem der Cursor aktuell positioniert ist). In der Regel wird die Ressource nicht direkt aufgerufen, sondern von der eigentlichen Funktion, die der Benutzer aufruft. Eine Ressource erhält die Eingabe also von der aufrufenden Emacs-Lisp-Funktion und liefert die Resultate an diese Funktion zurück.

Die Ressourcen sollten Programmierschnittstellen haben, um eine reibungslose Integration zu ermöglichen. Die Resultate sollten in einem Format geliefert werden, das eine Weiterverarbeitung ermöglicht, ohne dass sie aufwendig nachbearbeitet werden müssen. Wenn nur einzelne Elemente eines Resultates benötigt werden (z. B. lediglich die Angabe des Numerus einer morphologischen Analyse eines Substantivs), sollte der Zugriff darauf direkt möglich sein.

Geschwindigkeit und Abdeckung Die Ressourcen werden in Funktionen integriert, die vom Autor aufgerufen werden. Die meisten dieser Funktionen sind interaktiv, d. h., es werden Parameter übergeben oder spezifiziert, der Autor wählt aus Vorschlägen aus etc. Die Resultate einzelner Funktionen dienen wiederum als Eingabe für andere Funktionen. Eine Ressource, die Teil einer solchen Funktion ist, muss schnell aufgerufen und ausgeführt werden können. Benutzer sind bereit, lediglich wenige Sekunden auf die Rückmeldung einer Funktion zu warten. Wird die Aufgabe einer Funktion vom Benutzer als «einfach» eingeschätzt, reduziert sich die akzeptierte Wartezeit auf weit weniger als 2 Sekunden, siehe [Cooper et al. 2007,

Good 1981, Raskin 2000, Waloszek 2008a,b]. Die Ressourcen müssen möglichst alle Elemente, die gefunden werden sollen, auch finden. Die Abdeckung muss also sehr hoch sein.

Qualität der Resultate Die Akzeptanz einer Funktion, die das Schreiben erleichtern soll, hängt wesentlich von der Qualität der Resultate dieser Funktion ab. Die Qualität von Funktionen, die computerlinguistische Ressourcen verwenden, hängt wiederum von der Qualität der Resultate dieser Ressourcen ab.

Abhängig von der Art der sprachlichen Elemente, auf die sich eine Ressource bezieht, sollte diese jeweils möglichst alle intendierten Elemente erkennen bzw. bearbeiten (d. h., die Abdeckung muss möglichst hoch sein), und die Ergebnisse müssen möglichst korrekt sein (d. h., die Präzision muss möglichst hoch sein). Für Systeme, die in praxistaugliche Anwendungen integriert werden sollen, wird davon ausgegangen, dass sie für mehr als 90 % der Fälle die korrekten Angaben liefern[3] und keine falschen Ergebnisse erzeugen [Hull et al. 1987].

Abhängig von der inhärenten Ambiguität bestimmter sprachlicher Elemente hinsichtlich bestimmter Eigenschaften (z. B. die hohe morphosyntaktische Ambiguität von deutschen Substantiven, wie von Evert [2004] gezeigt), ist die Interaktion mit dem Benutzer notwendig. Es stellt sich dann jeweils die Frage, ob eine Funktion, die auf solchen Resultaten operiert und in jedem Fall die Interaktion mit dem Benutzer erfordert, tatsächlich den Schreibprozess erleichtert. Erscheint eine solche Funktion konzeptionell sinnvoll, kann sich herausstellen, dass sie aufgrund der Eigenschaften des Deutschen nicht implementiert werden sollte, da die notwendige Interaktion den eigentlichen Schreibprozess zu stark unterbricht und somit die kognitiven Anforderungen erhöht, statt diese zu mindern. Auf den Aspekt der Qualität von Resultaten gehen wir jeweils in den folgenden Abschnitten noch ein.

Wie in Abschnitt 4.3 gezeigt, ist der Einsatz computerlinguistischer Ressourcen abhängig vom gewünschten Funktionsumfang. In den folgenden Abschnitten betrachten wir Systeme zur morphologischen Analyse und Generierung von Wortformen, zur Bestimmung von Wortarten und zur flachen syntaktischen Analyse.

6.2 Morphologische Analyse und Generierung von Wortformen

Eine Funktion wie `query-replace-word` zur Ersetzung eines Wortes durch ein anderes im gesamten Text erfordert die Verwendung morphologischer Komponenten zur Analyse und Generierung von Wortformen. Diese Funktion ersetzt jedes Vorkommen des Suchwortes durch die entsprechende Wortform des Ersatzwortes unter Berücksichtigung der Kategorie der Vorkommen des Suchwortes – damit unterscheidet sich diese Funktion grundlegend von den üblichen `search & replace` Funktionen.

Im Folgenden formulieren wir die genauen Anforderungen für morphologische Systeme (Abschnitt 6.2.1 und 6.2.2), stellen die von uns betrachteten Systeme

3. Siehe z. B. die Aussage von Detmar Meurers während des Vortrag zu [Meurers et al. 2010] am 5. Juni 2010.

vor und evaluieren sie hinsichtlich der im vorigen Abschnitt vorgestellten einzelnen Anforderungen (Abschnitt 6.2.3 bis 6.2.7). Abschliessend begründen wir in Abschnitt 6.2.8, welches der Systeme wir auswählen.

6.2.1 Auswahlkriterien für betrachtete Systeme

Wie in vielen anderen Bereichen der Computerlinguistik, existieren sowohl regelbasierte als auch statistische Ansätze zur morphologischen Analyse und Generierung. Wir haben hier ausschliesslich regelbasierte Systeme berücksichtigt.

Zum einen basiert diese Entscheidung auf unseren eigenen Erfahrungen mit der Implementierung eines Systems zur morphologischen Analyse und der Gewissheit, dass diese detaillierte strukturierte Ergebnisse liefern können [vgl. Mahlow 2000, Mahlow und Piotrowski 2009a]. Es ist ein grosser Vorteil, explizit die morphologischen Prozesse Flexion, Derivation und Komposition abzubilden, auf entsprechende Ableitungsbäume und Zwischenergebnisse zugreifen und gezielt einzelne Elemente der Kategorie einer Wortform verwenden zu können. Diese Möglichkeiten bieten statistisch basierte Systeme in der Regel nicht.

Zum anderen – und dieser Aspekt gab den entscheidenden Ausschlag – sind die bislang verfügbaren statistischen Systeme zur morphologischen Analyse von deutschen Wortformen zu wenig überzeugend, um sie in praxistauglichen Anwendungen einzusetzen. Diese Einschätzung beruht auf den Ergebnisse von *Morpho Challenge*.

Morpho Challenge ist ein Wettbewerb (engl. *shared task*) zur Evaluation statistischer morphologischer Komponenten, die auf unüberwachtem maschinellen Lernen (engl. *unsupervised machine-learning*) basieren. Morpho Challenge wird seit 2005 durchgeführt. In der Aufgabe zur Analyse von Wortformen werden aktuell lediglich Wortformen in Morpheme zerlegt. Die Ergebnisse der beteiligten Systeme werden mit einem *Gold-Standard* verglichen. Im Wettbewerb von 2009 [Kurimo et al. 2009a,b], erreichte das beste System für Deutsch ein f-Mass (f-*measure*) von 56,14 %. Der beste Wert für Ausbeute (*recall*) war 99,92 % mit 2,79 % Präzision (*precision*), der beste Wert für Präzision war 81,70 % mit 22,98 % Ausbeute.[4] Diese Werte sind zu schlecht, um die entsprechenden Systeme tatsächlich in interaktiven Systemen einsetzen zu können. Es muss jedoch erwähnt werden, dass die beteiligten Systeme primär auf den Einsatz im Information Retrieval und in der maschinellen Übersetzung ausgerichtet sind.

Die verwendeten Texte für die einzelnen Aufgaben in Morpho Challenge stammen für Deutsch aus den Jahren 1994 und 1995: «…300K documents from short articles in Frankfurter Rundschau 1994, Der Spiegel 1994-95 and SDA German 1994-95, 60 test queries with 23K binary relevance assessments» [Kurimo et al. 2009b]. Die beteiligten Systeme sind für diese Texte optimiert, d. h., sie sind für deutschsprachige Texte in der Schreibung *vor* der Rechtschreibreform geeignet. Sie sind nicht geeignet für aktuell verfasste Texte – Schreibunterstützung für Autoren soll natürlich die aktuell gültige Rechtschreibung berücksichtigen. Insofern sind die an Morpho Challenge beteiligten Systeme nicht für

4. Siehe auch `http://www.cis.hut.fi/morphochallenge2009/` (zuletzt besucht am 8.12.2010, 19:34) für weitere Details. Die Resultate von Morpho Challenge 2010 wurden noch nicht publiziert.

einen realistischen Einsatz in anderen Systemen geeignet.[5] Da diese Texte ausschliesslich aus Zeitungen in Deutschland stammen, wird Schweizer Rechtschreibung (etwa «*gross*» statt «*groß*») und Wortbildung (etwa «*Zugsdurchfahrt*» statt «*Zugdurchfahrt*») ebenfalls nicht berücksichtigt.

Wir haben folgende regelbasierte Systeme in Betracht gezogen und evaluiert: (a) *Stripey Zebra* [Lorenz 1996, Schulze 2004], (b) *Morphisto* [Zielinski und Simon 2008, Zielinski et al. 2009], (c) *GERTWOL/GERGEN* [Haapalainen und Majorin 1994, Koskenniemi und Haapalainen 1996] und (d) *mOLIFde* [Clematide 2008]. Genauere Informationen zu den einzelnen Systemen werden in Abschnitt 6.2.3 gegeben.

GERTWOL/GERGEN ist ein System, das schon sehr lange verfügbar ist. Die Entwickler haben sich an der *Ersten Morpholympics* 1994 (siehe [Hausser 1996]) beteiligt. Es lag nahe, alle dort erfolgreichen Systeme in unserer Auswahl zu berücksichtigen. Nicht alle Systeme, die sich dort beteiligten, sind jedoch frei verfügbar, einige sind nur für bestimmte Betriebssysteme implementiert oder werden nicht weiter gepflegt. Darum wird etwa *Morphy* [Lezius 1996] nicht berücksichtigt. Die offiziell frei verfügbare aktuelle Version von *MORPH* [Hanrieder 1996] in *Java* als *ERmorph*[6] hat verschiedene Mängel, sodass die Entwickler von der Verwendung dieser Version abraten.[7] Die originale Implementierung in *Lisp* ist nicht verfügbar.

In der oben genannten Liste der berücksichtigten Systeme mag man *TAGH* [Geyken und Hanneforth 2006] vermissen, für das die Entwickler eine sehr hohe Erkennungsrate angeben. Leider können die Entwickler aber keine Aussage zur tatsächlichen Qualität der Ergebnisse machen: «Das TAGH-Morphologiesystem erreicht bei neueren Zeitungstexten eine Erkennungsrate von über 99 %. Die Genauigkeit der Analyse wurde bislang noch nicht systematisch ausgewertet.» [Geyken 2009]. *TAGH* steht in einer Demo-Version online zur Verfügung[8], die Ressourcen sind nicht frei verfügbar. Leider reagieren die Entwickler nicht auf Anfragen, das System in einer akademischen Lizenz zu erwerben, um es lokal zu installieren. Wir konnten *TAGH* darum nicht für unsere Evaluation bzw. das Projekt *LingURed* allgemein berücksichtigen.

Wir berücksichtigen ebenfalls nicht die aktuelle morphologische Komponente, die an der Abteilung Computerlinguistik der Friedrich-Alexander-Universität Erlangen-Nürnberg im Framework *JSLIM*[9] [Handl et al. 2009] entwickelt wird. Die Komponente für Deutsch ist zum Zeitpunkt des Verfassens dieser Arbeit noch nicht so mächtig, wie es das Vorgängersystem *DMM* war. Dies betrifft vor allem die Grösse des Lexikons. *JSLIM* ist zwar in der Lage, Wortformen zu generieren, andererseits können bislang nur Paradigmen bzw. einzelne Wortformen von Simplizia generiert werden, keine Derivationen, Komposita oder Verbgefüge.[10]

5. Wir werden in Abschnitt 6.3 feststellen, dass auch andere Ressourcen für computerlinguistische Systeme zur Behandlung von deutschsprachigen Texten für aktuelle Texte nicht geeignet sind.
6. Siehe `http://www8.informatik.uni-erlangen.de/inf8/de/demosdownloads.html` (zuletzt besucht am 17.12.2010, 14:51).
7. Persönliche Kommunikation mit Martin Hacker, 18. August 2010.
8. `http://www.tagh.de/` (zuletzt besucht am 8.12.2010, 19:34).
9. `http://www.linguistik.uni-erlangen.de/clue/de/forschung-projekte/jslim.html` (zuletzt besucht am 8.12.2010, 19:34).
10. Persönliche Kommunikation mit Johannes Handl (Entwickler), 6. September 2010.

Ebenfalls nicht berücksichtigt wurde *MPRO* [Maas 1996, Maas et al. 2009]. Die Aussagen über die Fähigkeit zur Generierung sind eher vage. Offenbar müssen bei *MPRO* Anpassungen an der Generierungskomponente vorgenommen werden.[11] Wegen der voraussagbaren Probleme im Bereich Generierung sowohl für *MPRO* als auch für *TAGH* wurden keine weiteren Anstrengungen unternommen, die Systeme zu installieren und zu testen.

6.2.2 Anforderungen

Die in Abschnitt 6.1 allgemein dargestellten Anforderungen gelten ohne Abstriche für morphologische Komponenten. Zum Aspekt der Qualität der Resultate geben wir hier noch spezifischere Kriterien an.

Damit eine morphologische Ressource möglichst alle Wortformen eines Textes analysieren kann, muss die Abdeckung möglichst gross sein. Diese ist hauptsächlich abhängig vom Lexikon. Für Wörter, die nicht im Lexikon enthalten sind, kann im günstigsten Fall eine Hypothese geliefert werden. Da Deutsch hinsichtlich Ableitungen und Komposita eine produktive Sprache ist, können natürlich nicht alle Wörter des Deutschen im Lexikon enthalten sein. Wenn die morphologische Analyse die Prinzipien von Derivation und Komposition abbildet, können jedoch «neue» Wörter erkannt und kategorisiert, d.h. hinsichtlich ihrer morphosyntaktischen Eigenschaften bestimmt werden.

Die Analyse einer Wortform soll möglichst korrekt und vollständig sein. Soweit möglich, sollten Ambiguitäten aufgelöst werden[12]; unplausible Zerlegungen und damit Rückführungen auf falsche Lemmata sollten vermieden werden.

Für die automatische Wortformanalyse erachten wir die von Hausser [2001] genannten Prozesse als relevant:

> Using the on-line lexicon, each unknown word form […] must be characterized automatically with respect to categorization and lemmatization:
>
> **Categorization** consists in specifying the part of speech […] and the morphosyntactic properties of the surface […]
>
> **Lemmatization** consists in relating a word form […] to the corresponding base form [the base form surface which is used as the key for storing or finding a lexical entry is called the *lemma*] […][Hausser 2001, 251f.]

Die Analyse einer Wortform enthält also das *Lemma* des Wortes und die *Kategorie* der Wortform – die Wortart und die morphosyntaktischen Eigenschaften. Diese explizite Nennung ist eigentlich trivial und entspricht eher der Verwendung innerhalb eines Lehrbuchs – als solches ist [Hausser 2001] auch konzipiert –, wir werden jedoch feststellen, dass diese grundlegenden Eigenschaften einer automatischen morphologischen Analyse nicht von allen aktuellen Systemen erfüllt werden.

11. Persönliche Kommunikation mit Christoph Rösener, E-Mail vom 23.7.2010, 9:45, Message-ID `<4C494896.3030107@iai.uni-sb.de>`.

12. Wie in [Mahlow und Piotrowski 2009a] argumentiert, können syntaktische und semantische Ambiguitäten durch morphologische Komponenten nicht aufgelöst werden, sie müssen erhalten bleiben. Systeme, die versuchen, entsprechende Ambiguitäten *während* der morphologischen Analyse aufzulösen, vermischen linguistische Ebenen und können nicht als morphologische Komponenten bezeichnet werden.

Für die Generierung muss es möglich sein, ausgehend von der Grundform eines Wortes (also ausgehend vom Lemma) entweder das gesamte Paradigma zu erhalten oder nach Angabe einer konkreten Kategorie die entsprechende Wortform (d. h. ihre Oberfläche). Für Verbgefüge wie «*aufschreiben*» erwarten wir idealerweise zwei Formen – diejenige, die in Hauptsätzen verwendet wird («*Er* ***schreibt*** *es* ***auf***.»), und diejenige, die in Nebensätzen verwendet wird («*[…], damit er es* ***aufschreibt***.»). Idealerweise ist es für die Generierung des Paradigmas eines Wortes nicht notwendig, das Lemma anzugeben, eine beliebige Wortform des Wortes sollte als Eingabe genügen.

Exkurs: Verbgefüge

Verbgefüge werden auch als *trennbare Verben* oder *Partikelverben* [Eisenberg 2006, S. 254ff.] bezeichnet. Neben *Partikelverben* existieren untrennbare *Präfixverben* [Eisenberg 2006, S. 246ff.], die aus einem Verbstamm und einem «Präfix» bestehen. Für Verbindungen wie «*versetzen*» oder «*belegen*» mag diese Bezeichnung zutreffen: Hier handelt es sich um eine Verbindung eines Verbs mit einem Präfix (etwa «*ver-*», «*be-*», «*ent-*» oder «*zer-*»). Solche Verben sind prinzipiell Derivationen wie «*rad-eln*» oder «*fest-igen*».

Schumacher et al. [2004, S. 48] unterscheiden *Simplexverben* und *Komplexverben*. Dieser Terminologie folgend, könnten Komplexverben in Ableitungen und Zusammensetzung unterteilt werden und letztere in fregesche – und damit in bestimmten syntaktischen Konstellationen als diskontinuierliche Konstruktion verwendbare – und nichtfregesche Komposita.[13] Schumacher et al. unterscheiden jedoch *komplexe Verben mit nicht-abtrennbarem Präfix* wie «*überfahren*» in «*Er überfährt die Linie*» von *Simplexverben mit einer Präposition* wie «*Er fährt über die Linie*» [Schumacher et al. 2004, S. 49].

Für «*ab*» in «*abstürzen*», «*über*» in «*übertreffen*» oder «*vor*» in «*vorstehen*» ist die Bezeichnung als Präfix jedoch falsch – häufig wird «*abstürzen*» als Verb mit abtrennbarem Präfix bezeichnet, «*übertreffen*» als Verb mit nichtabtrennbarem Präfix, «*vorstehen*» kann in beiden Varianten gebraucht werden. Die Bezeichnung *abtrennbares Präfix* oder *nichtabtrennbares Präfix* (siehe etwa [Schumacher et al. 2004, Zifonun et al. 1997]) ist irreführend. Korrekter wäre die Klassifizierung als Kompositum für Verben wie «*übertreffen*», ähnlich wie «*wetteifern*», «*schlussfolgern*», «*massregeln*», «*schlafwandeln*» oder «*brandmarken*», da die beteiligten Morpheme als eigenständige Wörter auftreten können und das resultierende Wort zusammengeschrieben wird. Es handelt sich nicht um die Modifikation des Verbs, sondern es ergibt sich eine ganz neue (idiomatische) Bedeutung, weswegen die Bestandteile nicht getrennt werden können. Es handelt sich also um nichtfregesche Komposita. Gehen die beteiligten Morpheme eine eher lockere (trennbare) Verbindung ein – das Verb wird modifiziert und in bestimmten syntaktischen Konstruktionen (Hauptsatz) wird diese Verbindung getrennt –, handelt es sich eher um ein syntaktisches als ein morphologisches Phänomen. Wir folgen hier der Argumentation von Donalies [2005, S. 28ff].

13. Ein Beispiel für ein nichtfregesches Kompositum ist «*Kirchenschiff*», das eben nicht ein «klerikales Wasserfahrzeug» bezeichnet.

Für Simplexverben, die mit einer Präposition verwendet werden, existiert in der englischen Linguistik die Bezeichnung *phrasal verb* [Quirk et al. 1985] – Verb und Präposition gehen eine sehr enge Verbindung ein, werden jedoch nicht als morphologisches, sondern syntaktisches Phänomen behandelt. In die gleiche Klasse gehören unserer Meinung nach Verbgefüge mit Verbzusatz, die in einigen syntaktischen Konstellationen zusammen, in anderen jedoch getrennt geschrieben werden. Da keine entsprechende deutsche Terminologie existiert, verwenden wir den Begriff *Verbgefüge*.[14]

Die Bezeichnung *Präverb* für die Nicht-Verb-Elemente in Verbgefügen, wie sie Donalies [2005] vorschlägt, ist irreführend, da in syntaktischen Konstellationen, die die Getrenntschreibung erfordern, das *Präverb* am Ende des Satzes steht, dem Verb also *nachfolgt* und nicht *vorausgeht*. Wir folgen daher der Terminologie vom Rat für Deutsche Rechtschreibung [2006] und betrachten untrennbare Verbkomposita wie «*übertreffen*» oder «*massregeln*» als Zusammensetzung aus Verbstamm mit vorausgehendem Substantiv-, Adjektiv- oder Präpositional- oder Partikelstamm. (Trennbare) *Verbgefüge* wie «*abstürzen*» bestehen aus einem Verbstamm mit *Verbzusatz* (Substantiv, Adjektiv, Partikel, Verb). [Rat für Deutsche Rechtschreibung 2006, S. 35f, §33f.][15]

Wichtig ist für unsere Zwecke, dass Konventionen es in einigen Fällen erfordern, Verbgefüge als ein Wort zu schreiben, während in anderen syntatktischen Konstellationen Verb und Verbzusatz getrennt geschrieben werden. Alle diese Varianten gehören zum Paradigma des Verbgefüges, darum erwarten wir von einer Generierungskomponente die Ausgabe sowohl der getrennt als auch der zusammen geschriebenen Wortformen.

Ende des Exkurses

Damit sind für Analyse und Generierung die Eingabeformate definiert:

- für die Analyse einer Wortform ist nur die Oberfläche dieser Wortform die Eingabe (z. B. «*Mütter*», «*Leistungsnachweises*», «*parkiert*»[16] oder «*hinaufgetragen*»)
- für die Generierung eines Paradigmas einer Wortform ist die Eingabe das Lemma (etwa «*Mutter*», «*Leistungsnachweis*», «*parkieren*», «*hinauftragen*») oder eine konkrete Wortform (analog der Eingabe für die Analyse)
- für die Generierung einer konkreten Wortform eines Wortes ist die Eingabe das Lemma und die gewünschte Kategorie

14. Die von uns so bezeichneten Verbgefüge, Derivationen und Komposita sind in der Linguistik kontrovers diskutierte Themen, zu denen bisher keine abschliessende Lösung festzustellen ist, vgl. [Donalies 1999, Knobloch 2009, Lüdeling 1999].

15. Wichtig: Wir gehen hier nicht auf die a. a. O. gemachten Überlegungen zur Zusammen- und Getrenntschreibung ein, sondern nur auf die verwendete Terminologie und die gegebenen Beispiele.

16. «*parkieren*» ist die schweizerische Variante von «*parken*», es handelt sich jedoch nicht um einen Dialektausdruck. Da wir diese Dissertation einerseits an einer Schweizer Universität verfassen und andererseits das Schreiben «deutscher» Texte auch regionale Varianten einschliesst, müssen solche Varianten auch von den verwendeten Ressourcen behandelt werden können.

(etwa `Mutter N DAT PL`, `Leistungsnachweis N GEN SG`, `parkieren V IND PRÄS PL 2`, `hinauftragen V PART`)

Wir verwenden die angegebenen Beispiele für die Evaluation der Systeme. Daneben testen wir alle Systeme auf ihre Geschwindigkeit und die Abdeckung für die Analyse umfangreicher Texte. Die morphologische Komponente soll in eine Funktion integriert werden. Um die Zeit für die Ausführung dieser Funktion möglichst gering zu halten, sollte keine Nachbearbeitung der von der morphologischen Ressource gelieferten Resultate notwendig sein.

Für morphologische Komponenten erachten wir die Kriterien (a) Qualität der Resultate, (b) Schnittstellen und Format der Resultate und (c) Geschwindigkeit und Abdeckung als entscheidend (*must have*). Systeme, die bezüglich dieser Kriterien schlechte Ergebnisse aufweisen, werden nicht berücksichtigt. Die Kriterien (d) Installation/Kompilieren und (e) Verfügbarkeit sollten möglichst gut erfüllt werden, hier sind jedoch auch Einschränkungen möglich (*nice to have*). Die ausgewählte Morphologiekomponente sollte in möglichst vielen Kriterien die besten Werte erreichen.

6.2.3 Verfügbarkeit und Installation

6.2.3.1 Stripey Zebra

Stripey Zebra ist die aktuelle Version der Morphologiekomponente für Deutsch, die ursprünglich als *Deutsche Malaga-Morphologie* (DMM) [Lorenz 1996] an der Abteilung Computerlinguistik der Friedrich-Alexander-Universität Erlangen-Nürnberg seit Mitte der 1990er Jahre entwickelt wurde. *Stripey Zebra* basiert auf dem Framework *Malaga*, entwickelt von Björn Beutel[17]. *Malaga* selbst ist eine von mehreren Implementierungen der *Linksassoziativen Grammatik* (LAG) [Hausser 2001].

Die von uns verwendete *Malaga*-Version ist 7.12. *Malaga* ist frei unter einer GNU General Public License (GPL) verfügbar und lässt sich mit den üblichen Kommandos «`configure`, `make`, `make install`» für verschiedene Betriebssysteme installieren. Wir verwenden ein *MacBook* mit *Mac OS X 10.5.4*, 2.16 GHz Intel Core 2 Duo und 2 GB 667 MHz DDR2 SDRAM, jedoch haben wir Malaga und darauf basierende Morphologiekomponenten auch auf verschiedenen Versionen von Solaris, HP-UX, NetBSD und Linux problemlos installiert.

Für die morphologische Analyse wird ein Allomorph-Lexikon verwendet, das vor der Laufzeit aus einem Grundformlexikon über Allomorphieregeln generiert wird. Dieser Prozess ist sehr schnell und wenig rechenintensiv. *Stripey Zebra*, d. h. das Lexikon und die Regeln, steht uns in einer Forschungslizenz in kompilierter Form zur Verfügung.[18] Das Lexikon enthält etwa 50'200 Grundformeinträge.[19] Das Lexikon kann mit beliebigen Einträgen erweitert werden.

17. `http://home.arcor.de/bjoern-beutel/malaga/` (zuletzt besucht am 8.12.2010, 19:34).
18. Wir haben also keinen direkten Zugriff auf Lexikon und Regeln, das Allomorphlexikon ist bereits erzeugt. Daher können wir die entsprechenden Daten zur Erzeugung des Lexikons nicht angeben.
19. Dieser Wert basiert auf der *DMM*-Version 5.0, dem Vorgänger von *Stripey Zebra* 1.1.

6.2.3.2 Morphisto

Morphisto wurde am Institut für Deutsche Sprache (IDS), Mannheim, innerhalb des TextGrid Projekts[20] entwickelt und ist eine der neueren Entwicklungen im Bereich morphologischer Analyse und Generierung für Deutsch. Es ist keine vollständige morphologische Komponente, sondern ein Open-Source Lexikon und eine Reihe von Verbesserungen für die *SMOR*-Morphologie [Schmid et al. 2004]. *SMOR* wiederum basiert auf dem Stuttgart Finite State Transducer Tool (*SFST*)[21]. Wir bezeichnen die Kombination aus *SFST*, *SMOR* und dem *Morphisto*-Lexikon als *Morphisto*.

Für unsere Experimente verwendeten wir *SFST* 1.3, das unter GPL frei verfügbar ist. *SMOR* ist in der *SFST* Distribution enthalten und als endlicher Automat implementiert. Wir konnten *SFST* unter *Mac OS X*, *NetBSD* und *Linux* installieren. Benötigt wird zusätzlich *Python* in einer Version ab 2.4 und *libxml2* sowie *xsltproc*. Alle benötigten Ressourcen sind frei verfügbar, die Hardware- und Softwareanforderungen für die Installierung sind nicht sehr hoch. Anschliessend installierten wir den *Morphisto*-Release vom 12. Dezember 2008, ebenfalls frei erhältlich[22] unter einer *Creative Commons*-Lizenz «Attribution-NonCommercial 2.0». Dadurch werden einige *SMOR*-Regeln überschrieben und das sehr kleine *SMOR*-Lexikon wird durch ein umfangreicheres Lexikon (19'234 Einträge) ersetzt.

Zuerst müssen die Transducer kompiliert werden, die dieses Lexikon verwenden. Auf der *Morphisto*-Webseite wird dazu ein Computer mit *mindestens* 8 GB RAM empfohlen. Unsere Erfahrungen bestätigen dies: Die Kompilierung des Tranducers für die Analyse benötigte mehr als zweieinhalb Stunden auf einem Linux-Server[23]. Das Kompilieren des invertierten Transducers für die Generierung dauerte genauso lange. Das anschliessende Komprimieren der Transducer (um sie schneller ausführen zu können) benötigte dann nur noch sechs Sekunden. Insgesamt werden also mehr als fünf Stunden und ein Computer mit ausreichend RAM benötigt, um die gesamte Komponente kompilieren zu können. Die Transducer (jeweils etwa 23 GB gross) können anschliessend auch auf Systemen mit weniger RAM ausgeführt werden.

Das *Morphisto*-Lexikon in der Version vom 12. Dezember 2008 enthält 19'234 Einträge, die sich entsprechend Tabelle 6.1 zusammensetzen:

Tabelle 6.1: Zusammensetzung des Lexikons von Morphisto*, aufgeschlüsselt nach Wortarten.*

Wortart	Kürzel	Anzahl Lemmata
Substantive	NN	9'855
Verben	V	4'741
Adjektive	ADJ	3'467
Adverben	ADV	768
Andere		403

20. http://www.textgrid.de/ (zuletzt besucht am 8.12.2010, 19:34).
21. http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/SFST.html (zuletzt besucht am 8.12.2010, 19:34).
22. Erhältlich von http://ids-mannheim.de/ll/TextGrid/morphisto.html (zuletzt besucht am 8.12.2010, 19:34).
23. 2 2.5 GHz Dual Core Opteron Prozessor, 8 GB RAM, *Ubuntu 8.04*. Die Kapazität des MacBook reichte nicht aus.

6.2.3.3 GERTWOL/GERGEN

GERTWOL (wie auch die Generierungskomponente *GERGEN*) ist ein kommerzielles Produkt[24] der Firma Lingsoft Oy, Helsinki[25]. Es wird seit Mitte der 1980er Jahre entwickelt. *GERTWOL* für die Analyse und *GERGEN* für die Generierung sind im Binärfomat für verschiedene Plattformen erhältlich. Sie basieren auf endlichen Automaten und dem Two-Level-Ansatz von Koskenniemi [1983]. Eine frühere Version von *GERTWOL* war 1994 der Gewinner der ersten *Morpholympics*, siehe [Hausser 1996]. *GERTWOL* wurde als Ressource für Rechtschreibprüfungen entwickelt – Lingsoft Oy fertigt die deutschsprachigen Überprüfungskomponenten, die in den *Microsoft-Office*-Programmen enthalten sind. Zudem können entsprechende Komponenten für *Adobe*-Produkte und *QuarkXPress* erworben werden.

Für unsere Experimente verwenden wir die *GERTWOL*-Version vom 15. Juli 2010. *GERTWOL* und *GERGEN* bestehen aus einer *shared library* (die vermutlich eine Version der *finite-state-engine* von *Xerox* enthält) und plattformunabhängigen Lexikondaten. Das aktuelle Lexikon enthält über 150'000 Einträge. Laut Dokumentation sind verschiedene Varianten des Lexikons erhältlich, die den verschiedenen Schreibkonventionen entsprechen. Unsere Version ist diejenige entsprechend der Rechtschreibreform von 2006 ohne Berücksichtigung der Schweizer Rechtschreibung.

Die shared library lässt sich in verschiedene Applikationen einbinden. Für die Evaluation verwenden wir kleinere C-Programme, die Wortformen von der Standardeingabe lesen und die Ergebnisse auf der Standardausgabe ausgeben.

6.2.3.4 mOLIFde

mOLIFde ist eine experimentelle Morphologiekomponente, die am Institut für Computerlinguistik der Universität Zürich entwickelt wird. Sie basiert auf dem *Xerox Finite-State Tool* (*XFST*) [Karttunen et al. 1998] und einem Lexikon im *Open Lexicon Interchange Format* (OLIF)[26]. Es ist geplant, das *mOLIFde*-Lexikon und die Regeln unter einer Open-Source-Lizenz zu veröffentlichen.

XFST ist nicht frei erhältlich; mit dem Erwerb des Buches von Beesley und Karttunen [2003] erhält man eine Lizenz für *XFST*. Updates stehen anschliessend kostenlos zur Verfügung. *XFST* wird in Binärform für verschiedene Plattformen ausgeliefert. Wir konnten es für verschiedene Versionen von *Mac OS X*, *Solaris* und *Linux* installieren. Das `lookup`-Programm der von uns verwendeten *XFST*-Distribution ist die Version 2.3.0 (8.0.5).

Um den Transducer mit dem *mOLIFde*-Lexikon zu kompilieren, benötigt man auf einem *MacBook* mit *Mac OS X 10.5.4*, 2.16 GHz Intel Core 2 Duo und 2 GB 667 MHz DDR2 SDRAM etwa 10 Minuten. Der Transducer wird sowohl für die Analyse als auch für die Generierung verwendet.

Das Lexikon enthält nur Substantive, Verben und Adjektive, keine Vertreter nichtflektierender Wortarten, keine Artikel und Pronomina und keine Affixe, diese sollen später ergänzt werden. Über Regeln werden Ableitungen aus dem

24. Es ist möglich, akademische Lizenzen zu einem erschwinglichen Preis zu erwerben.
25. `http://lingsoft.fi/` (zuletzt besucht am 8.12.2010, 19:34).
26. `http://www.olif.net/` (zuletzt besucht am 8.12.2010, 19:34).

Grundlagenlexikon (38'228 Wortformen) erzeugt, die anschliessend getestet werden, ob sie in Korpora belegt sind. Daraus resultiert das sehr grosse Grundformenlexikon. Die Version vom 21. Juli 2008 umfasst insgesamt 139'996 Einträge, die sich entsprechend Tabelle 6.2 zusammensetzen. Komposita werden nicht vorab erzeugt, sondern während der Analyse über Regeln erkannt.

Tabelle 6.2: Zusammensetzung des Lexikons von mOLIFde*, aufgeschlüsselt nach Wortarten.*[27]

Wortart	Lexikon	Konversion	Derivation	insgesamt
Substantive	20'532	21'987	15'526	58'045
Verben	5'522	0	17'393	22'915
Adjektive	12'174	43'865	2'997	59'036

6.2.3.5 Zusammenfassung

Die berücksichtigten Systeme *Stripey Zebra*, *Morphisto*, *GERTWOL/GERGEN* und *mOLIFde* sind für verschiedene Plattformen erhältlich – eine sehr vielversprechende Situation. Allerdings ist nur ein System vollständig als Open-Source-System erhältlich: *Morphisto*. Die Kompilierung von *Morphisto* stellt leider erhebliche Anforderungen an die Hardware und dauert sehr lang.

6.2.4 Programmierschnittstellen, Format der Resultate und Möglichkeiten zur Weiterverarbeitung

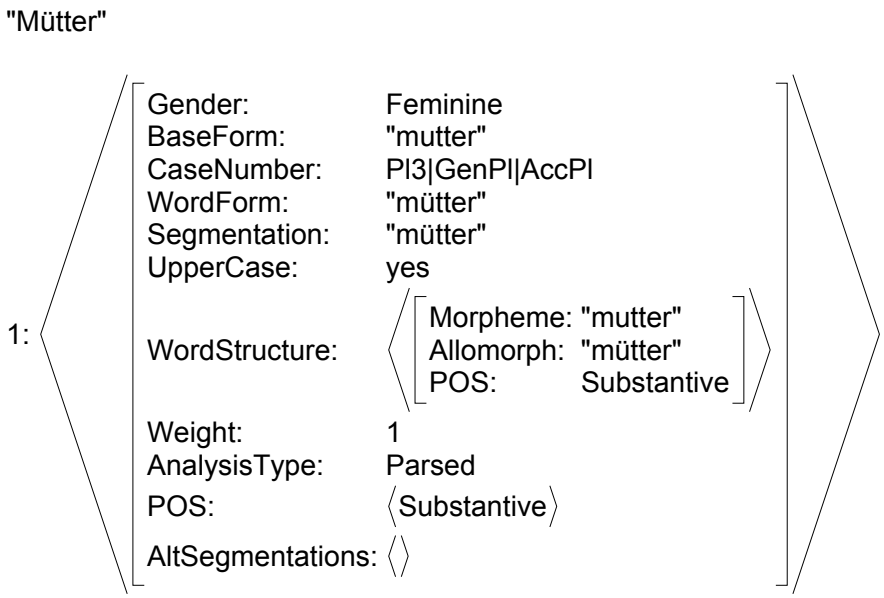
Wir betrachten zunächst die Analyse von Wortformen und stellen die Analyseergebnisse für «*erschien*», «*Mütter*», «*Leistungsnachweises*», «*parkiert*» und «*hinaufgetragen*» für jedes der vier Systeme dar. Um die morphologische Komponente in eine Funktion integrieren zu können, sollten die Analyseresultate in einer Form vorliegen, die es erlaubt, möglichst schnell und möglichst gezielt auf einzelne Informationen zugreifen zu können.

6.2.4.1 Stripey Zebra

Malaga stellt interaktive und batch-basierte Funktionen zur Verfügung sowie eine C-Library. Es existieren Module für *Perl*, *Ruby* und *Python*, die über eine Schnittstelle eingebunden werden können. So ist es möglich, *Stripey Zebra* aus einem Programm heraus aufzurufen, Wortformen zur Analyse zu übergeben und anschliessend auf einzelne Werte der Analyse zuzugreifen, um diese weiterzuverarbeiten. Die Analysen werden in distinktiver Form[28] als hierarchische Attribut-Werte-Strukturen zurückgeliefert, wie in Abbildung 6.1 auf der nächsten Seite für die Analyse von «*Mütter*» dargestellt. Die Abbildung zeigt die graphische Darstellung. Daneben ist auch eine textbasierte Ausgabe möglich, wie in Listing 6.1 auf der nächsten Seite. Diese Ausgabe kann für weitere Applikationen und Prozesse verwendet werden. Die Datenstruktur ermöglicht den gezielten Zugriff auf Werte bestimmter Attribute.

27. Konversion: Partizipien von Verben können als Substantiv oder Adjektiv verwendet werden; Derivation: Mittels Präfix und/oder Suffix; insgesamt: Gesamtzahl der Grundformen dieser Wortart.
28. Siehe [Hausser 2001, S. 244, S. 346] für die Unterschiede zwischen distinktiver und exhaustiver Kategorisierung.

Abbildung 6.1: Stripey Zebra*: Analyse von* «Mütter» *(graphisch mit eingeschaltetem Ausgabefilter).*



Listing 6.1: Stripey Zebra*: Analyse von* «Mütter» *(textbasiert mit eingeschaltetem Ausgabefilter).*

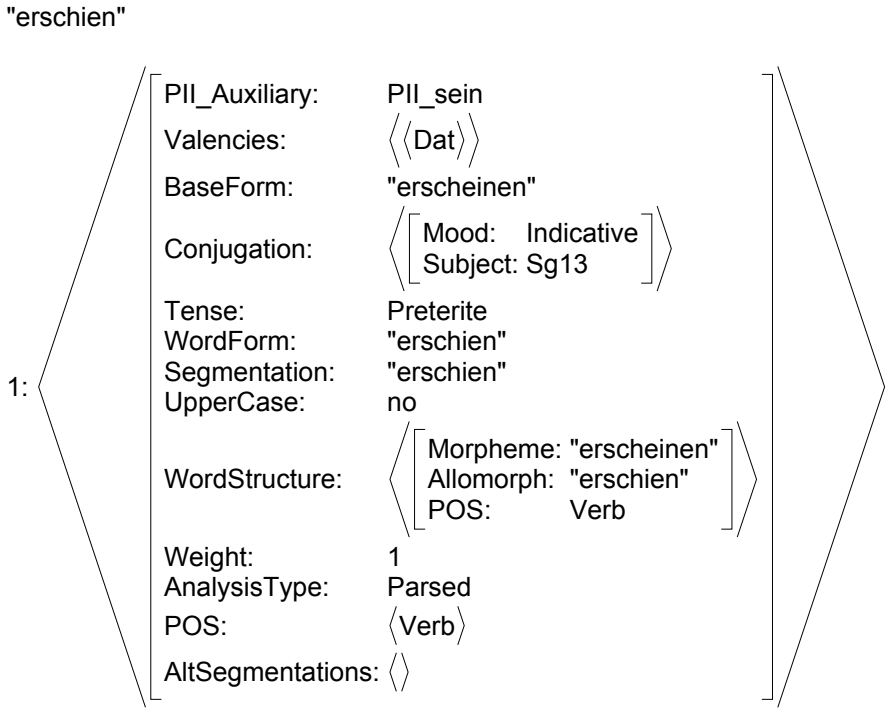
```
Analyses of "Mütter":
1: <[Gender: Feminine,
     BaseForm: "mutter",
     CaseNumber: Pl3|GenPl|AccPl,
     WordForm: "mütter",
     Segmentation: "mütter",
     UpperCase: yes,
     WordStructure: <[Morpheme: "mutter",
                      Allomorph: "mütter",
                      POS: Substantive]>,
     Weight: 1,
     AnalysisType: Parsed,
     POS: <Substantive>,
     AltSegmentations: <>]>
```

Die Analyse liefert keine explizite Angabe zur Flexionsklasse. Diese kann jedoch aus den Werten der entsprechenden Attribute rekonstruiert werden. Es kann direkt zugegriffen werden auf: (a) das Lemma (`BaseForm`), (b) die Kategorie (hier die Wortart `POS` sowie `Gender` und `CaseNumber`) sowie (c) die Segmentierung (`Segmentation` und `WordStructure`, letztere enthält zusätzliche Informationen zu den beteiligten Morphemen). Die Analyse entspricht unseren Bedingungen: Die Eingabe ist lediglich die zu analysierende Wortform, das Ergebnis enthält die relevanten Angaben zu Lemma und Kategorie dieser Wortform.

Abbildung 6.2 auf der nächsten Seite für die Analyse von «*erschien*» zeigt alle Daten, die während der Analyse erzeugt, in der Regel jedoch für eine Weiterverarbeitung nicht benötigt werden. Über Ausgabefilter können diese Angaben unterdrückt werden.

Malaga erlaubt es, Regeln für die Behandlung unbekannter und mit den eigentlichen Regeln nicht behandelbarer Wortformen zu erstellen. Diese «robust»-Regeln können für die Ermittlung hypothetischer Kategorien verwendet werden. Zusätzlich verwendet *Stripey Zebra* Gewichtung. Die Konkatenation der ermittelten Allomorphe einer Wortform erhält jeweils ein Gewicht. Aus allen Gewichten wird ein Gesamtgewicht der Analyse erreichnet. Hierdurch können unwahrscheinliche Segmentierungen und damit fehlerhafte Analysen ausge-

Abbildung 6.2: Stripey Zebra*: Analyse von* «erschien» *(graphisch ohne Ausgabefilter).*



schlossen werden, da sie ein geringeres Gewicht erhalten. Die Verwendung von Gewichtung wird ebenfalls über den Ausgabefilter gesteuert.

6.2.4.2 Morphisto

Als Schnittstellen für *Morphisto* stehen die Werkzeuge von *SFST* interaktiv oder batch-basiert zur Verfügung. Es ist möglich, auf die Ergebnisse der Analyse und Generierung zuzugreifen, da die Ausgaben jeweils textbasiert sind und über reguläre Ausdrücke die benötigten Informationen leicht extrahiert werden können. Es existiert ein *Python*-Modul und der Aufruf von *Morphisto* kann über Shell-Skripte gesteuert werden.

Listing 6.2: Morphisto*: Analyse von* «Mütter».

```
> Mütter
Mutter<+NN><Fem><Gen><Pl>
Mutter<+NN><Fem><Nom><Pl>
Mutter<+NN><Fem><Akk><Pl>
```

Listing 6.3: Morphisto*: Analyse von* «erschien».

```
> erschien
erscheinen<+V><1><Sg><Past><Ind>
erscheinen<+V><3><Sg><Past><Ind>
```

Die Analyse einer Wortform besteht aus einer Liste der verschiedenen Lesarten und zugehörigen Kategorien. Dies entspricht einer exhaustiven Kategorisierung. Eine Kategorie ist ein String, keine Datenstruktur. Für «*Mütter*» ist die Kategorie als `<+NN><Fem><Gen><Pl>` angegeben (siehe Listing 6.2), für «*erschien*» ist sie `<+V><3><Sg><Pres><Ind>` (siehe Listing 6.3). Es ist nicht möglich, direkt auf den Wert für ein bestimmtes Attribut – etwa Numerus – zuzugreifen, da es keine fixe Position für diese Angabe gibt: Für Substantive ist der Numerus die dritte bzw. letzte Angabe, für Verben die zweite nach der Angabe für die Wortart.

Auch wenn im Webinterface[29] und in *TextGrid*[30] von «lemmatisieren» die Rede ist, liefert diese Ausgabe *nicht* das Lemma der analysierten Wortform. Auf den ersten Blick mag dies so scheinen, die Angabe `Mutter` in Listing 6.2 auf der vorherigen Seite oder die Angabe `erscheinen` in Listing 6.3 auf der vorherigen Seite ist jedoch nur die Angabe zur Segmentierung. Diese Tatsache wird bei der Analyse komplexerer Wortformen deutlich, siehe Listing 6.4. Es wird lediglich die Segmentierung der analysierten Wortform mit der Kennzeichnung der Wortart der beteiligten Segmente und der insgesamt resultierenden Wortart sowie die Kategorie der Wortform zurückgeliefert.

Listing 6.4: Morphisto*: Analyse von* «Leistungsnachweises».

```
> Leistungsnachweises
Leistung<NN>Nachweis<+NN><Masc><Gen><Sg>
leisten<V>ung<NN><SUFF>Nachweis<+NN><Masc><Gen><Sg>
```

Vergleichen wir die zweite Analyse von «*Leistungsnachweises*» in Listing 6.4 mit der von *Stripey Zebra* gelieferten in Abbildung 6.3, wird deutlich, dass die Segmente jeweils die Grundformen der beteiligten Morpheme sind und die Wortart des jeweiligen Morphems angegeben wird. Die gleichen Informationen finden sich in der Ausgabe des Analyseergebnisses von *Stripey Zebra* als `WordStructure`. An diesem Beispiel wird ebenfalls deutlich, dass *Morphisto* auf die Angabe von Fugenelementen verzichtet.

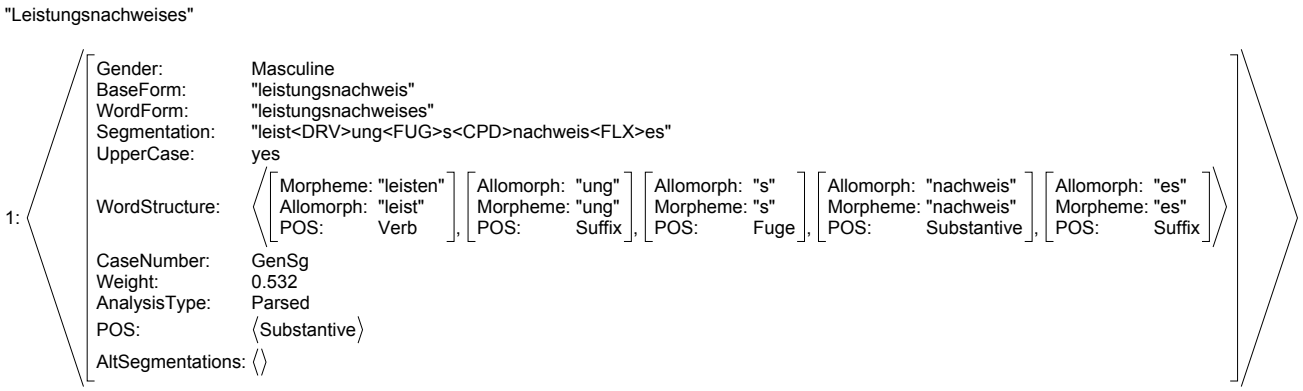


Abbildung 6.3: Stripey Zebra*: Analyse von* «Leistungsnachweises» *(graphisch).*

Aus der Angabe der segmentierten Wortform lässt sich das Lemma somit nicht ohne Weiteres ableiten, ein Zugriff auf das Lemma ist also nicht möglich. Damit werden die elementarsten Anforderungen an eine Komponente zur automatischen morphologischen Analyse nicht erfüllt.

Morphisto ist damit kein «Lemmatisierer», obwohl es im *TextGrid*-Projekt als solcher entwickelt wurde und bezeichnet wird. Die Funktionsbeschreibung innerhalb von *TextGrid* lautet:

> Ein Lemmatisierer für das Deutsche (morphologische Analyse) analysiert einzelne Wortformen (Tokens) und liefert als Ausgabe das zugehörige Lemma (d. h. das Token wird auf seine grammatische Grundform zurückgeführt), die zugehörige Wortart (PartOfSpeech) und weitere morphologische Merkmale (Numerus, Genus usw.).[31]

29. Siehe http://ingrid.sub.uni-goettingen.de/cgi-bin/analyze.cgi (zuletzt besucht am 8.12.2010, 19:34).
30. Siehe http://www.textgrid.de/beta.html (zuletzt besucht am 8.12.2010, 19:34).
31. http://www.textgrid.de/beta.html (zuletzt besucht am 8.12.2010, 19:34).

Tatsächlich wird *entweder* die Analyse (in der hier gezeigten Form) geliefert *oder* das «Lemma» – interessanterweise wird diese Funktionalität als `Get Wordform` bezeichnet.

Für die Abfragemöglichkeit im Internet und für die Einbindung in *TextGrid* wird ein Python-Skript verwendet, das eine Funktion `beautify` enthält – diese Funktion soll das Lemma für eine Wortform liefern. Wie oben schon gezeigt, ist es nicht möglich, aus den Angaben einer Analyse einer Wortform mit einfachen Zeichenkettenoperationen das Lemma zu extrahieren. `beautify` verwendet die Grundform, Angaben aus der Analyse und einige Heuristiken, um ein «Lemma» zu produzieren. So wird für «*geteilt*» etwa «*geteilen*» als Lemma ermittelt, für «*aufgeteilt*» jedoch korrekt «*aufteilen*».

6.2.4.3 GERTWOL

GERTWOL wird als *shared library* mit einer C-Schnittstelle ausgeliefert. Die Entwickler stellen ein Beispielprogramm zur Verfügung, es existieren jedoch keine Standardfunktionen wie etwa für *SFST*. Es ist daher notwendig, selbst Funktionen zur Verwendung von *GERTWOL* zu implementieren.

Jede Analyse besteht aus der Angabe der Grundform mit der Segmentierung – d. h. der Kennzeichnung, ob Derivation, Komposition, Fugenelemente etc. für die Bildung der Grundform verwendet werden – und einer Liste von Tags. Auch hier wird das Lemma nicht explizit zurückgeliefert. Es ist jedoch möglich, aus der segmentierten Wortform das Lemma zu extrahieren, indem die Segmentierungssymbole entfernt werden. Die Kategorie einer Wortform ist eine Liste von Tags, wobei jede Position in dieser Liste einem grammatikalischen Attribut entspricht; die 1. Position ist immer die Wortart, die 6. Position ist der Kasus – wird dieses Attribut für eine Wortform nicht ermittelt (etwa für Verben), wird ein leerer Wert zurückgeliefert.

Listing 6.5 und 6.6 zeigen die Analysen für «*Mütter*» und «*erschien*»; leere Werte sind als «–» gekennzeichnet. Damit lässt sich aus der Position der Werte, die nicht «–» sind, die Kategorie einer Wortform darstellen und es kann explizit auf bestimmte Werte zugegriffen werden.[32]

Listing 6.5: GERTWOL*: Analyse von* «Mütter».

```
Mütter
(
("Mutter" . [N - FEM PL - NOM - - - - - - - - -])
("Mutter" . [N - FEM PL - ACC - - - - - - - - -])
("Mutter" . [N - FEM PL - GEN - - - - - - - - -])
)
```

Listing 6.6: GERTWOL*: Analyse von* «erschien».

```
erschien
(
("er|schein~en" . [V - - - - - - - - PAST IND - - - SG1 -])
("er|schein~en" . [V - - - - - - - - PAST IND - - - SG3 -])
("er|schien~en" . [V - - - - - - - - PRES IMP - - - SG2 -])
)
```

Frühere Versionen von *GERTWOL*[33] lieferten in der Analyse für die Kategorie lediglich eine Zeichenkette, die alle relevanten Angaben enthielt. Diese

32. Aus Gründen der Lesbarkeit zeigen wir in den folgenden Analysen von *GERTWOL* nicht in allen Fällen die leeren Werte.
33. Uns steht die Version vom 5. Juni 2000 zur Verfügung.

Ausgabe ist ähnlich wie die von *Morphisto* nicht geeignet, gezielt auf Werte bestimmter Attribute zuzugreifen. Es ist möglich, diese Variante der Analyse (als *LSLING* bezeichnet) neben derjenigen mit Listen von Tags (als *LSINDEX* bezeichnet) zu erhalten. Die *LSLING* Kategorie ist jeweils als Tag mit dem Index 0 in der *LSINDEX*-Ausgabe enthalten.

Eine weitere Änderung betrifft die verwendeten Tags für Werte bestimmter Attribute: Diese sind nun englisch (wie für alle anderen Systeme auch), während sie vordem deutsch waren (und in der *LSINDEX*-Angabe weiterhin sind). Diese Änderung ist jedoch eher «kosmetischer» Art und hat keinen Einfluss auf die Analysen selbst. Listing 6.7 zeigt die Analyse von «*hinaufgetragen*», wie sie die *GERTWOL*-Version aus dem Jahr 2000 liefert.

Listing 6.7: GERTWOL *2000: Analyse von* «hinaufgetragen».

```
"<hinaufgetragen>"
        "hinauf|trag~en"  V TRENNBAR PART PERF
        "hin|auf|ge|tragen"  A(PART) POS
        "hin|auf|trag~en"  V TRENNBAR PART PERF
        "hinauf|ge|tragen"  A(PART) POS
```

Vergleicht man diese Analyse mit der Analyse, die die 2010-Version liefert, fallen mehrere (negative) Veränderungen auf. Listing 6.8 zeigt die Ausgabe entsprechend *LSLING* und die Ausgabe entsprechend *LSINDEX*.

Listing 6.8: GERTWOL *2010: Analyse von* «hinaufgetragen».

```
hinaufgetragen
"<hinaufgetragen>"
        "hin|auf|trag~en"       V TRENNBAR PART PERF
        "hin|auf|ge|trag~en"    A(PART) POS
(
("hin|auf|trag~en" .    [V PTC2])
("hin|auf|ge|trag~en" . [POS])
)
```

Die *LSLING*- und die *LSINDEX*-Ausgabe stimmen in der Segmentierung überein. Mit *LSINDEX* werden jedoch nur noch zwei statt ehemals vier Analysen (siehe Listing 6.7) angegeben. Die zwei Analysen, die nicht mehr ausgegeben werden, unterscheiden sich in der Behandlung des Verbzusatzes: «*hinauf*» wird in die Bestandteile «*hin*» und «*auf*» zerlegt, die Analysen, die «*hinauf*» als eigenständigen Verbzusatz behandeln, entfallen.

In der tag-basierten Ausgabe ist nun zudem keine Angabe mehr darüber enthalten, dass «*hinauftragen*» ein Verbgefüge, d. h. Verbbasis mit Verbzusatz, ist. Die Information als solche ist zwar vorhanden, denn sie wird in der *LSLING*-Variante weiterhin ausgegeben – als `TRENNBAR` –, in der tag-basierten Ausgabe wird sie dann jedoch unterdrückt. Eine weitere unterdrückte Information ist die Angabe `A` als Wortart für «*hinaufgetragen*». In der *LSINDEX*-Ausgabe ist keine Wortart angegeben, sie muss durch die Angabe `POS` (Positiv) erschlossen werden. Der Hinweis, dass es sich um ein adjektivisch gebrauchtes Partizip `(PART)` handelt, ist ebenso nicht mehr sichtbar, gehört allerdings auch nicht zu den morphosyntaktischen Eigenschaften im engeren Sinne.[34]

Als Fehler ist zu betrachten, dass der Verbzusatz in «*hinauf*» nicht explizit angegeben wird – die Analyse lässt den Schluss zu, dass «*hin*» der Verbzusatz sei, was falsch ist.

34. Wir gehen in Abschnitt 6.2.6.3 und 6.2.7.3 noch näher auf unterdrückte Informationen in der *LSINDEX*-Ausgabe ein.

6.2.4.4 mOLIFde

Definierte Programmierschnittstellen zur Weiterverarbeitung der Ergebnisse der Analyse einer Wortform, der Analyse einer Wortformliste oder des generierten Paradigmas eines Wortes existieren für *mOLIFde* nicht. Da der Aufruf von Analyse und Generierung jeweils über Shell-Skripte erfolgt, die die entsprechenden Werkzeuge von *XFST* aufrufen, und die Ausgabe textbasiert ist, kann einerseits der Aufruf interaktiv eingebunden werden und andererseits können die Ergebnisse gut weiterverarbeitet werden. *mOLIFde* ist also prinzipiell für den interaktiven Einsatz geeignet.

Problematisch ist die Erzeugung einer sehr grossen Zahl von Ableitungen aus dem Grundformlexikon, siehe Tabelle 6.2 auf Seite 133, wodurch sich die Anzahl der Lexikoneinträge mehr als verdreifacht. Diese Ableitungen sind morphologisch korrekt, aber eventuell nicht gebräuchlich, darum wird mittels Korpusanalyse versucht, ihre Verwendung nachzuweisen. Sprache entwickelt sich jedoch ständig weiter. Es ist durchaus vorstellbar, dass zu einem späteren Zeitpunkt mehr Ableitungen in dann aktuellen Korpora nachgewiesen werden können; das Lexikon muss also ständig geprüft werden. Die Tatsache, dass eine erzeugte Wortform in Korpora gefunden wird, mag genügen, um zu entscheiden, dass es sich um eine gültige Wortform handelt. Die Tatsache, dass eine bestimmte Wortform *nicht* in Korpora gefunden wird, kann jedoch nicht als Kriterium dafür verwendet werden, die Gültigkeit dieser Wortform zu verneinen.

mOLIFde hat grosse Schwierigkeiten mit der Analyse komplexer Komposita. Die Wortform «*Leistungsnachweises*» wird nicht erkannt, dies gilt für das gesamte Paradigma von «*Leistungsnachweis*». Analysiert man eine Version dieses Kompositums ohne Fugenelement, also «*Leistungnachweises*», werden Analysen zurückgeliefert – nur ist diese Wortform nicht wohlgeformt.

Listing 6.9: mOLIFde*: Analyse von* «Mütter».

```
Mütter   53      Mutter   NN              Fem.Nom.Pl.*
Mütter   53      Mutter   NN              Fem.Akk.Pl.*
Mütter   53      Mutter   NN              Fem.Gen.Pl.*
```

Die Analyse für eine Wortform enthält Angaben zu *OLIF*-Flexionscode (kann unterdrückt werden), Grundform, Wortart und Kategorie, siehe Listing 6.9 für die Analyse von «*Mütter*». Das Format der Analyseresultate ist ähnlich dem von *Morphisto*. Das Lemma kann jedoch aus dieser Ausgabe rekonstruiert werden, da Fugenelemente mit aufgeführt werden. *mOLIFde* erfüllt also durch zusätzliche einfache Nachbearbeitung der Ergebnisse (ähnlich denen für *GERTWOL*) die elementaren Anforderungen an Komponenten zur automatischen morphologischen Analyse. Ob für komplexe, mehrfach abgeleitete und zusätzlich durch Komposition gebildete Wortformen ebenfalls das Lemma einfach rekonstruiert werden kann, konnte nicht ermittelt werden, da *mOLIFde* für solche Wortformen keine Analyse liefert.

Die Wortart einer Wortform wird über automatische Wortartenbestimmung mittels *TreeTagger* [Schmid 1994] ermittelt, siehe [Clematide 2008]. Für jede Wortform wird so zunächst die Wortart bestimmt und erst dann werden die spezifischen *mOLIFde*-Regeln angewendet. *TreeTagger* liefert zwar sehr gute Resultate, diese sind jedoch nicht vollständig korrekt, hier liegt also bereits eine Fehlerquelle. Für Wortfomen, die über die *mOLIFde*-Regeln nicht analysiert

werden können, wird in jedem Fall die von *TreeTagger* bestimmte Wortart ausgegeben (siehe die Listings 6.11 bis 6.14).

6.2.4.5 Zusammenfassung

In den Listings 6.10, 6.11, 6.12, 6.13 und 6.14 auf den nächsten Seiten zeigen wir zusammengestellt die Analysen der Systeme für die Wortformen «*Mütter*», «*Leistungsnachweises*», «*parkiert*», «*hinaufgetragen*» und «*erschien*». Für *Stripey Zebra* wurde jeweils die Version mit Filter eingesetzt, für *GERTWOL* verwenden wir die *LSINDEX*-Ausgabe.

Die Ergebnisse von *Stripey Zebra* und *GERTWOL* sind direkt vergleichbar: Die ermittelte Segmentierung komplexer Wortformen, die ermittelten Kategorien und die Anzahl der Analysen (unter Berücksichtigung exhaustiver und distinktiver Kategorisierung) sind sehr ähnlich. Fugenelemente werden erkannt und in der Analyse markiert. *Stripey Zebra* erfüllt als einziges System die Anforderung, Lemma und Kategorie explizit zurückzuliefern. Zudem sind die Analysen von *Stripey Zebra* am detailliertesten und bieten die meisten Möglichkeiten für eine Einbindung in andere Applikationen. Für Wortformen, die von *Stripey Zebra* nicht über die eigentlichen Regeln erkannt werden können, werden Hypothesen geliefert, aus denen die meisten notwendigen Informationen gewonnen werden können (siehe die Analyse von «*parkiert*» in Listing 6.12 auf Seite 143). *Morphisto* liefert leider kein Lemma; *mOLIF* kann komplexe Wortformen (d. h. Komposita und abgeleitete Wortformen) nicht analysieren. Für beide kann nicht gezielt auf Werte bestimmter Attribute zugegriffen werden.

Listing 6.10: Analysen von «Mütter».

```
## Stripey Zebra ##

Analyses of "Mütter":
1: <[Gender: Feminine,
     BaseForm: "mutter",
     CaseNumber: Pl3|GenPl|AccPl,
     WordForm: "mütter",
     Segmentation: "mütter",
     UpperCase: yes,
     WordStructure: <[Morpheme: "mutter",
                      Allomorph: "mütter",
                      POS: Substantive]>,
     Weight: 1,
     AnalysisType: Parsed,
     POS: <Substantive>,
     AltSegmentations: <>]>

## Morphisto ##

> Mütter
Mutter<+NN><Fem><Nom><Pl>
Mutter<+NN><Fem><Gen><Pl>
Mutter<+NN><Fem><Akk><Pl>

## GERTWOL ##

Mütter
(
("Mutter" . [N - FEM PL - NOM - - - - - - - - -])
("Mutter" . [N - FEM PL - ACC - - - - - - - - -])
("Mutter" . [N - FEM PL - GEN - - - - - - - - -])
)

## mOLIFde ##

Mütter  NN      Mutter  Fem.Nom.Pl.*
Mütter  NN      Mutter  Fem.Akk.Pl.*
Mütter  NN      Mutter  Fem.Gen.Pl.*
```

Listing 6.11: Analysen von «Leistungsnachweises».

```
## Stripey Zebra ##

Analyses of "Leistungsnachweises":
1: <[Gender: Masculine,
     BaseForm: "leistungsnachweis",
     WordForm: "leistungsnachweises",
     Segmentation: "leist<DRV>ung<FUG>s<CPD>nachweis<FLX>es",
     UpperCase: yes,
     WordStructure: <[Morpheme: "leisten",
                      Allomorph: "leist",
                      POS: Verb],
                     [Allomorph: "ung",
                      Morpheme: "ung",
                      POS: Suffix],
                     [Allomorph: "s",
                      Morpheme: "s",
                      POS: Fuge],
                     [Allomorph: "nachweis",
                      Morpheme: "nachweis",
                      POS: Substantive],
                     [Allomorph: "es",
                      Morpheme: "es",
                      POS: Suffix]>,
     CaseNumber: GenSg,
     Weight: 0.532,
     AnalysisType: Parsed,
     POS: <Substantive>,
     AltSegmentations: <>]>

## Morphisto ##

> Leistungsnachweises
Leistung<NN>Nachweis<+NN><Masc><Gen><Sg>
leisten<V>ung<NN><SUFF>Nachweis<+NN><Masc><Gen><Sg>

## GERTWOL ##

Leistungsnachweises
(
("Leist~ung\s#nach|weis" . [N - MASC SG - GEN - - - - - - - - -])
)

## mOLIFde ##

Leistungsnachweises     PDAT
```

Listing 6.12: Analysen von «parkiert».

```
## Stripey Zebra ##

Analyses of "parkiert":
1: <[POS: <PII>,
     WordForm: "parkiert",
     AnalysisType: Hypothesis,
     Weight: 0.01,
     BaseForm: "parkiert",
     Segmentation: "parkiert",
     WordStructure: <[Morpheme: "parkiert",
                      Allomorph: "parkiert"]>,
     StemHypothesis: [Stem: "parkieren",
                      StemPOS: Verb],
     AltSegmentations: <>],
    [POS: <Verb>,
     Conjugation: <[Mood: Indicative,
                    Subject: Sg3|Pl2]>,
     WordForm: "parkiert",
     AnalysisType: Hypothesis,
     Weight: 0.01,
     BaseForm: "parkier",
     Segmentation: "parkier<FLX>t",
     WordStructure: <[Morpheme: "parkier",
                      Allomorph: "parkier"],
                     [Morpheme: "t",
                      Allomorph: "t"]>,
     StemHypothesis: [Stem: "parkieren",
                      StemPOS: Verb],
     AltSegmentations: <>]>

## Morphisto ##

> parkiert
no result for parkiert

## GERTWOL ##

parkiert
(
("park~ier~en" . [V - - - - - -   - PRES IND -    - - PL2 -])
("park~ier~en" . [V - - - - - -   - PRES IMP -    - - PL2 -])
("park~ier~en" . [V - - - - - -   - PRES IND -    - - SG3 -])
("park~ier~en" . [V - - - - - -   - -    -   PTC2 - - -   -])
("park~ier~t" .  [- - - - - - POS - -    -   -    - - -   -])
)

## mOLIFde ##

parkiert        ADJD
```

Listing 6.13: Analysen von «hinaufgetragen».

```
## Stripey Zebra ##

Analyses of "hinaufgetragen":
1: <[BaseForm: "hinauftragen",
     WordForm: "hinaufgetragen",
     Segmentation: "hinauf<PFX>ge<CPD>trag<GZE>en",
     UpperCase: no,
     WordStructure: <[Morpheme: "hinauf",
                      Allomorph: "hinauf",
                      POS: Prefix],
                     [Allomorph: "ge",
                      Morpheme: "ge",
                      POS: IVCS],
                     [Allomorph: "trag",
                      Morpheme: "tragen",
                      POS: Verb],
                     [Allomorph: "en",
                      Morpheme: "en",
                      POS: Suffix]>,
     PII_Auxiliary: PII_haben,
     Valencies: <<Acc>,
                 <Acc, DatBen>>,
     PII: yes,
     Weight: 1,
     AnalysisType: Parsed,
     POS: <PII>,
     AltSegmentations: <>]>

## Morphisto ##

> hinaufgetragen
hinauf<PREF>tragen<V><PPast><SUFF><+ADJ><Pos><Adv>
hinauf<PREF>tragen<V><PPast><SUFF><+ADJ><Pos><Pred>
hinauf<PREF>tragen<+V><PPast>

## GERTWOL ##

hinaufgetragen
(
("hin|auf|trag~en" .    [V - - - - - -   - - - PTC2 - - - -])
("hin|auf|ge|trag~en" . [- - - - - - POS - - - -    - - - -])
)

## mOLIFde ##

hinaufgetragen  VVPP
```

Listing 6.14: Analysen von «erschien».

```
## Stripey Zebra ##

Analyses of "erschien":
1: [PII_Ge: no,
    PII_Auxiliary: PII_sein,
    Valencies: <<Dat>>,
    InflexType: strong,
    StemForm: SF34,
    concatSx: yes,
    BaseForm: "erscheinen",
    PhonEnd: unmarked,
    Conjugation: <[Mood: Indicative,
                   Subject: Sg13]>,
    terminal: yes,
    Tense: Preterite,
    concatStem: no,
    WordForm: "erschien",
    Segmentation: "erschien",
    UpperCase: no,
    WordStructure: <[Morpheme: "erscheinen",
                     Allomorph: "erschien",
                     POS: Verb]>,
    LastLexeme: Verb,
    Weight: 1,
    AnalysisType: Parsed,
    POS: <Verb>]

## Morphisto ##

> erschien
erscheinen<+V><1><Sg><Past><Ind>
erscheinen<+V><3><Sg><Past><Ind>

## GERTWOL ##

erschien
(
("er|schein~en" . [V - - - - - - - PAST IND - - - SG1 -])
("er|schein~en" . [V - - - - - - - PAST IND - - - SG3 -])
("er|schien~en" . [V - - - - - - - PRES IMP - - - SG2 -])
)

## mOLIFde ##

erschien        VVFIN
```

6.2.5 Geschwindigkeit und Abdeckung

Tabelle 6.3 zeigt die Ergebnisse eines Experiments zur Analyse mit *Stripey Zebra*, *Morphisto*, *GERTWOL* und *mOLIFde*. Dafür wurden alle Systeme auf Texte der Zeitung «Der Tagesspiegel»[35] aus dem Zeitraum 2005/2006 mit 2'183'441 laufenden Wortformen[36] in 133'056 Sätzen angewandt. Diese Texte entsprechen alle der aktuell gültigen Rechtschreibung (d. h. den Regeln der Reform von 1996, die 2004 und 2006 noch einmal überarbeitet wurden [Rat für Deutsche Rechtschreibung 2006]). Zeitungstexte decken inhaltlich sehr viele verschiedene Gebiete ab, sodass der verwendete Wortschatz sehr gross ist. In Zeitungstexten werden häufig auch «neue» Wörter verwendet, die sich mit den Regeln zur Derivation und Komposition analysieren lassen, jedoch (noch) in keinem Lexikon vorhanden sind. Dieser Aspekt ist für uns interessant, da Schreiben einen kreativen Aspekt enthält: Autoren «spielen» mit Wörtern und Sätzen, sie «erfinden» neue Wörter – unterstützende Funktionen sollen darum auch diese bislang unbekannten Wörter behandeln können.

	2'183'441 laufende Wortformen					**179'644 distinkte Wortformen**				
	WF analys.	% analys.	An. je WF	Zeit (s)	WF/s	WF analys.	% analys.	An. je WF	Zeit (s)	WF/s
GERTWOL	2'131'929	97,64	5,79	497	4'393	153'379	85,38	6,59	62	2'897
Morphisto	2'062'893	94,48	5,50	55	39'698	142'220	79,17	9,58	8	22'455
SZ o. Gw.	2'095'743	95,98	**1,86**	490	4'456	142'315	79,22	**2,57**	85	2'113
SZ m. Gw.	2'183'441	100,00	**1,00**	533	4'096	179'644	100,00	**1,00**	95	1'890
mOLIFde	693'609	**31,77**	4,26	99	22'085	36'313	**20,21**	5,41	12	15'152

Tabelle 6.3: Ergebnisse der Systeme für die «Tagesspiegel»-Texte.[37]

Die Analysen wurden alle auf einem *MacBook* mit *Mac OS X 10.6.3*, 2.4 GHz Intel Core 2 Duo und 4 GB 1067 MHz DDR3 SDRAM durchgeführt. Bezüglich der Geschwindigkeit und Abdeckung waren wir interessiert an (a) der Erkennungsrate (d. h. dem Verhältnis von analysierten und nicht analysierten Wortformen), (b) der Anzahl Analysen je Wortform und (c) der benötigten Zeit, um das komplette Korpus bzw. die Liste der distinkten Wortformen zu analysieren. Daraus ergibt sich die Anzahl von analysierten Wortformen je Sekunde. Die Korrektheit und Vollständigkeit der Analysen betrachten wir in einem gesonderten Experiment in Abschnitt 6.2.6.

Für *Stripey Zebra* führten wir zwei Analysen durch: einmal mit Gewichtung und unter Verwendung der Robustheitsregeln (so werden nur die Analysen mit der höchsten Wahrscheinlichkeit zurückgeliefert und für alle unbekannten Wortformen werden Hypothesen gebildet, daraus resultiert eine Erkennungsrate von 100 %) und einmal ohne diese.

35. «Der Tagesspiegel. Zeitung für Berlin und Deutschland», Verleger: Dieter von Holtzbrinck, Herausgeber: Dr. Pierre Gerckens, Giovanni di Lorenzo, Dr. Hermann Rudolph, Verlag Der Tagesspiegel GmbH, Berlin.

36. Wir unterscheiden *laufende Wortformen* (engl. *running word forms*) und *distinkte Wortformen* (engl. *unique word forms*).

37. Abkürzungen: *WF analys.*: Anzahl analysierter Wortformen; *% analys.*: Prozentsatz analysierter Wortformen; *An. je WF*: Durchschnittliche Anzahl Analysen je analysierter Wortform; *Zeit (s)*: benötigte Zeit für die Analyse in Sekunden, *WF/s*: Anzahl behandelter Wortformen je Sekunde. *SZ o.Gw.*: *Stripey Zebra* ohne «Gewichtung» und «Robustheit»; *SZ m. Gw.*: *Stripey Zebra* mit «Gewichtung» und «Robustheit».

Die Resultate enthalten zwei überraschende Werte: die sehr geringe Erkennungsrate von *mOLIFde* und die sehr geringe Anzahl von Analysen je Wortform von *Stripey Zebra*.

Eine Erklärung für die geringe Erkennungsrate von *mOLIFde* ist die Tatsache, dass lediglich Verben, Substantive und Adjektive analysiert werden können und dass das System noch im Entwicklungsstadium ist. Zudem wird vor der eigentlichen Analyse eine automatische Wortartenbestimmung mit *TreeTagger* durchgeführt. Die Wortformen, deren Wortart so nicht oder falsch erkannt wird, beeinflussen die Fehlerrate. Komplexe Komposita, für die beispielsweise *Morphisto* sehr viele Analysen liefert, werden von *mOLIFde* nicht erkannt. Da jedoch eine Abdeckung von weniger als 50 % viel zu gering für eine praxistaugliche Anwendung ist, wurde *mOLIFde* nicht in die weiteren Experimente zur Qualität der Analysen einbezogen und für *LingURed* nicht berücksichtigt.

Die geringe Zahl von Analysen je Wortform von *Stripey Zebra* kann mit dem Design des Systems erklärt werden. Die Verwendung von Gewichtung bewirkt, dass nur die wahrscheinlichste Analyse als Ergebnis ausgegeben wird. Zudem verwendet *Stripey Zebra* distinktive Kategorisierung, während die anderen Systeme exhaustive Kategorisierung verwenden. Aus diesem Grund ist die Anzahl der Analysen je Wortform nicht über alle Systeme hinweg vergleichbar.

Die Erkennungsquote von GERTWOL bestätigt die sehr guten Werte, die im Rahmen der *Morpholympics* 1994 ermittelt wurden, siehe [Hausser 1996, S. 13f] und [Koskenniemi und Haapalainen 1996]. In diesem Zusammenhang ist erstaunlich, dass die neueren Systeme die Werte von *GERTWOL* hinsichtlich der Anzahl erkannter Wortformen nicht übertreffen.

Der allgemeine Eindruck von *Stripey Zebra*, *Morphisto* und *GERTWOL* bezüglich Geschwindigkeit und Abdeckung ist gut. *Morphisto* ist schneller als *Stripey Zebra* und *GERTWOL*, hat jedoch eine leicht geringere Erkennungsrate. Wie in Tabelle 6.3 auf der vorherigen Seite dargestellt, liegt die Anzahl der Wortformen, für die eine Analyse ermittelt wird, für alle drei Systeme über 94 % für laufende Wortformen und über 79 % für distinkte Wortformen. Die Anzahl der Analysen je Wortform ist dagegen unterschiedlich; die Verwendung von distinktiven Kategorien durch *Stripey Zebra* und die Verwendung von exhaustiven Kategorien durch *Morphisto* und *GERTWOL* erklärt nur einige dieser Unterschiede.

6.2.6 Qualität der Analysen

Für *Stripey Zebra*, *Morphisto* und *GERTWOL* wurde die Qualität der gelieferten Analysen ermittelt. Es ist nicht möglich, die ermittelten Daten mit denen der Systeme zu vergleichen, die an der ersten Morpholympics 1994 teilnahmen, da die Qualität der Analysen dort nicht ermittelt wurde [vgl. Lenders et al. 1996, S. 14].

Werden alle Wortformen eines Textes morphologisch analysiert, kann die Qualität der morphologischen Komponente bestimmt werden, indem untersucht wird, wieviele der Wortformen vollständig korrekt analysiert werden. Unter *vollständig korrekt* verstehen wir, dass *alle* der gelieferten Analysen für eine Wortform richtig sind und keine Analyse fehlt. Eine vollständige Analyse besteht aus dem Lemma und der Kategorie der Wortform (siehe Abschnitt 6.2.2), zusätzli-

che Informationen, wie Zerlegung in Allomorphe, Angaben von Verbvalenzen etc., werden nicht bewertet.

Für ein entsprechendes Experiment haben wir eine Stichprobe von 384 Wortformen aus dem Tagesspiegelkorpus (genauer: aus der Wortliste der distinkten Wortformen) ausgewählt. Die Stichprobengrösse gewährleistet ein Konfidenzniveau von 95 % mit 5 % Stichprobenfehler, entsprechend der Standardformel

$$n = \frac{Z^2 \sigma^2}{e^2} \tag{6.1}$$

$Z^2 = 1.96$ ermöglicht ein Konfidenzniveau von 95 %, e ist der gewünschte Präzisionsgrad (wir verwenden ein Konfidenzinterval von 5 %) und σ^2 ist die Varianz der Attribute der Gesamtmenge (wir verwenden $\sigma^2 = 0,25$ für maximale Variabilität).

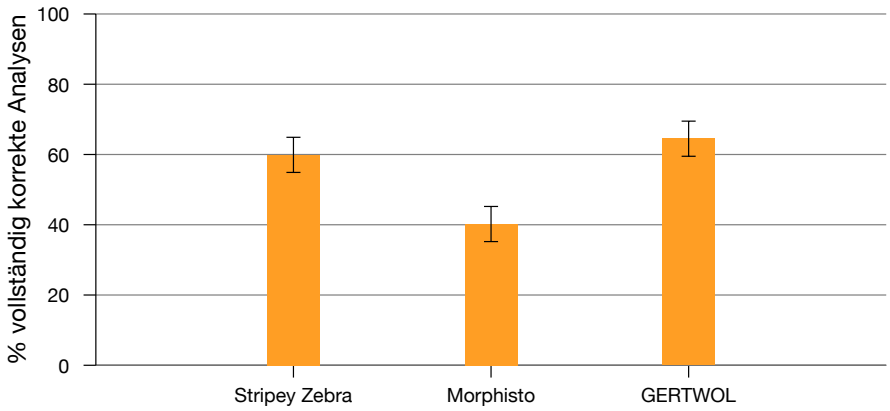


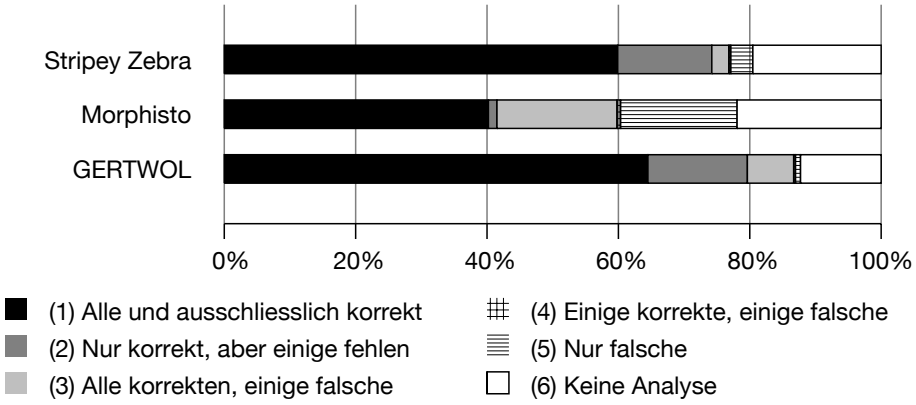
Abbildung 6.4: Prozentsatz der vollständig korrekten Analysen mit einem Konfidenzintervall von 5 %.

Diese 384 Wortformen wurden durch jedes der drei Systeme analysiert und die Analysen manuell durch einen – d. h. denselben – Annotator evaluiert. Die Ergebnisse sind in Abbildung 6.4 dargestellt. Wir haben die Analysen nicht automatisch gegen einen Gold-Standard evaluiert. Die Unterschiedlichkeit der Formate der Resultate der einzelnen Systeme hätte es erfordert, diese in ein einheitliches Format zu überführen. Die manuelle Evaluation war hier die günstigere Lösung.

Die Anzahl der «erkannten» Wortformen – 79 bis 85 % für distinkte Wortformen – ist sehr viel höher als die Anzahl der korrekt analysierten Wortformen – diese beträgt nur rund 60 %. *Stripey Zebra* (59,90 %) und *GERTWOL* (64,50 %) liefern ähnliche Werte, beide sind besser als *Morphisto*, welches nur 40,21 % vollständig korrekte Analysen ermittelt – dieser Unterschied zwischen den Systemen hinsichtlich vollständig korrekter Analyen ist grösser als der hinsichtlich erkannter Wortformen (vgl. Tabelle 6.3 auf Seite 146). Die Anzahl der erkannten Wortformen unserer Stichprobe entspricht den ermittelten Werten beim Performanz-Test, unsere Stichprobe ist also repräsentativ. Vergleichen wir diese Zahlen mit denen entsprechender Experimente für ein Korpus der «Neuen Zürcher Zeitung» und das Limas-Korpus [siehe Mahlow und Piotrowski 2009c], stellen wir lediglich für *Morphisto* eine grössere Abweichung fest – für das NZZ-Korpus und das Limas-Korpus waren die Analysen von *Morphisto* zu über 50 % korrekt. Wir gehen später noch auf diese Unterschiede ein.

Betrachtet man die Analysen genauer, lassen sich weitere Unterschiede zwischen den Systemen feststellen. Abbildung 6.5 zeigt die detaillierten Ergebnisse der Evaluation. Die Analysen wurden entsprechend der folgenden Skala bewertet: (1) alle und ausschliesslich korrekte Analysen der Wortform werden zurückgeliefert[38]; (2) nur korrekte Analysen werden zurückgeliefert, sie sind jedoch nicht vollständig; (3) alle korrekten Analysen werden zurückgeliefert, jedoch auch einige falsche; (4) einige korrekte Analysen fehlen, es werden auch falsche Analysen zurückgeliefert; (5) alle zurückgelieferten Analysen sind falsch; (6) es wird keine Analyse ermittelt. Im Folgenden diskutieren wir die Ergebnisse für jedes der drei Systeme.

Abbildung 6.5: Detaillierte Auswertung der morphologischen Analysen.



6.2.6.1 Stripey Zebra

Wir verwenden *Stripey Zebra* mit Gewichtung, jedoch ohne Robustheitsregeln. So werden keine Hypothesen für unbekannte Wortformen erstellt und es wird in der Regel nur eine Analyse zurückgeliefert. Es war daher zu erwarten, dass wenige Ergebnisse mit 3 (alle korrekten Analysen sind vorhanden, jedoch auch falsche) und 4 (einige korrekte und einige falsche Analysen) bewertet würden.

Abbildung 6.6: Stripey Zebra*: Analyse von* «unzugängliches» *(graphisch mit Gewichtung).*

"unzugängliches"

1:
- AdverbType: Modal
- BaseForm: "unzugänglich"
- Comparation: Positive
- WordForm: "unzugängliches"
- Segmentation: "unzugänglich<FLX>es"
- UpperCase: no
- WordStructure: ⟨ [Morpheme: "unzugänglich", Allomorph: "unzugänglich", POS: Adjective], [Allomorph: "es", Morpheme: "es", POS: Suffix] ⟩
- AdjDeclension: es
- Weight: 1
- AnalysisType: Parsed
- POS: ⟨DeclAdjective⟩
- AltSegmentations: ⟨"un<CPD>zugänglich<FLX>es"⟩

Relativ viele Analysen wurden mit 2 (nur korrekte Analysen, jedoch unvollständig) bewertet. Der Hauptgrund dafür liegt im Format der Resultate für flektierte Adjektive wie «*unzugängliches*», siehe Abbildung 6.6. Die Analyse liefert den Wert `DeclAdjective` als Wortart, es handelt sich also um ein flektiertes Adjektiv. Zusätzlich erscheint das Attribut `AdjDeclension` mit einem

38. Diese Werte korrespondieren also mit denen in Abbildung 6.4 auf der vorherigen Seite.

entsprechenden Wert (`es`). Es ist jedoch keine explizite Information zu Genus, Kasus, Numerus und bezüglich der Flexionsart (stark, schwach, gemischt) enthalten. Diese Informationen können aus dem Wert für `AdjDeclension` erschlossen werden, hierzu ist jedoch ein Nachbearbeitungsschritt notwendig.

Es mag sehr streng erscheinen, diese Analysen mit 2 zu bewerten. Da wir jedoch in Abschnitt 6.2.2 als Anforderung explizit gemacht haben, dass wir möglichst keine Nachbearbeitung der Resultate der morphologischen Ressource vornehmen möchten, ist diese Bewertung gerechtfertigt. Würde uns *Stripey Zebra* inklusive aller Ressourcen zur Verfügung stehen, könnte die Ausgabe der Analysen für flektierte Adjektive mit sehr geringem Aufwand verändert werden, sodass die benötigten Informationen explizit in der Analyse erscheinen. Da dies nicht der Fall ist, ist ein Nachbearbeitungsschritt notwendig, der jedoch über ein Perl-Skript sehr einfach und schnell erfolgen kann.

In der Stichprobe gibt es – wie erwartet – kaum Analysen, die mit 3 und 4 bewertet wurden. Die Anzahl der Analysen, die mit 5 (nur falsche Analysen) bewertet wurden, ist ebenfalls sehr gering. Einige der mit 5 (nur falsche Analysen) bewerteten Analysen sind falsch, da die Gewichtung die korrekten Analysen unterdrückt. Dazu gehören Pluralformen von Substantivierungen auf «*ung*», wie «*Marktstimmungen*», das als Kompositum aus «*Markt-stimm-un-gen*» und damit als Substantiv, Neutrum, Singular Nominativ/Dativ/Akkusativ analysiert wird – betrachtet man alle Analysen dieser Wortform ohne Gewichtung, ist dort auch die korrekte Analyse als Kompositum aus «*Markt-stimm-ung-en*» vorhanden. Dagegen wird «*Abschlussverhandlungen*» korrekt analysiert, d. h., die Gewichtung ist hier korrekt. Ähnlich ist es bei «*Bundesvermögensamt*»: Die Gewichtung für «*Bund-es-vermögen-samt*» ist höher als die für «*Bund-es-vermögen-s-amt*».

Würde die Ausgabe für flektierte Adjektive verbessert, stiege die Anzahl der vollständig korrekt analysierten Wortformen an und erreichte fast den Wert für insgesamt analysierte Wortformen.

6.2.6.2 Morphisto

Der Vergleich der Anzahl der Analysen je Wortform von *Stripey Zebra* und *Morphisto* ist nicht ohne Weiteres möglich. Die Analyse von «*Mütter*» für *Morphisto* ergib drei Analysen, im Gegensatz zu nur einer Analyse, die von *Stripey Zebra* geliefert wird (siehe Listing 6.10 auf Seite 141). Der Unterschied in der Anzahl der Analysen ist in der Art der Analysen bzw. im Design der Systeme begründet: *Stripey Zebra* verwendet distinktive, *Morphisto* verwendet exhaustive Kategorisierung. Alle Kategorien, die *Morphisto* für «*Mütter*» liefert, sind in der einen Analyse von *Stripey Zebra* enthalten, die Analysen sind also hinsichtlich der Kategorisierung äquivalent.

Nur sehr wenige Analysen von *Morphisto* werden mit 2 (nur korrekte Analysen, jedoch unvollständig) bewertet. Die Anzahl von Analysen, die mit 3 (einige korrekte, einige falsche Analysen) und 5 (nur falsche Analysen) bewertet wurden, ist jedoch sehr hoch. Der Hauptgrund für diese Bewertung liegt darin, dass *Morphisto* – oder vielmehr die zugrundeliegende *SMOR* Komponente – kein Lemma liefert. Die Analyse einer Wortform besteht aus der Angabe der morphosyntaktischen Eigenschaften und einem String, der die Segmentierung der Wortform enthält, wie in Abschnitt 6.2.4.2 beschrieben. Es ist nicht möglich,

durch einfache Zeichenkettenoperationen daraus das Lemma zu rekonstruieren, etwa für «*Jahrespressekonferenz*» oder «*Vermittlerrolle*»:

```
vermitteln<V>er<NN><SUFF>Rolle: NN Fem Nom Sg
Jahr<NN>Presse<NN>Konferenz: NN Fem Nom Sg
```

Die Mängel von *Stripey Zebra* können durch Expandieren der Werte für die Adjektivflexionsklassen einfach behoben werden. Das Lemma kann für die Analysen von *Morphisto* jedoch nur mit Hilfe von linguistischem Wissen ermittelt werden, es reicht beispielsweise nicht aus, die eingeschobenen Tags zu entfernen. Da *Morphisto* die elementaren Anforderungen an ein System zur morphologischen Analyse nicht erfüllt, müssten alle Analysen als komplett falsch bewertet werden. Wir waren jedoch etwas nachsichtiger und haben (analog zu *GERTWOL*, siehe nächster Abschnitt) aus der Zeichenkette mit der Segmentierung alle Segmentierungsinformationen entfernt und die resultierende Zeichenkette als Lemma bewertet.

Könnte für *Morphisto* tatsächlich eine Grundform geliefert werden, würden sehr viel weniger falsche Analysen zurückgeliefert werden.

6.2.6.3 GERTWOL

Wie in Abschnitt 6.2.4.3 bereits angesprochen, ist die Qualität der aktuellen Version von *GERTWOL* schlechter als die der Version von 2000. Um trotzdem möglichst korrekte Analysen zu erhalten, verwenden wir die Ausgaben entsprechend *LSLING*, die im Gegensatz zu *LSINDEX* keine Informationen unterdrückt. Sonst würden einige Analysen fälschlicherweise mit 2 (nur korrekte, jedoch unvollständig) bewertet. Allerdings verzichten wir so auf die Möglichkeit, direkt auf den Wert für eine bestimmte morphosyntaktische Eigenschaft (etwa Genus) zugreifen zu können.

Wie für *Stripey Zebra* wird ein relativ grosser Anteil der Analysen der Stichprobe mit 2 bewertet. Problematisch sind wiederum flektierte Adjektive. Anders als *Stripey Zebra*, liefert *GERTWOL* vollständige Kategorien für flektierte Adjektive inklusive Angabe, ob es sich um starke oder schwache Deklination handelt. Jedoch wird die gemischte Deklination nicht berücksichtigt. Damit fehlen für alle Wortformen eines Adjektivs, die auch der gemischten Deklination zugeordnet werden, entsprechende Angaben. Dieses Problem liesse sich mit einer Erweiterung des Systems leicht beheben. Da wir jedoch nicht über die Ressourcen verfügen – *GERTWOL* steht uns nur in einer kommerziellen Version zur Verfügung –, müsste auch hier eine entsprechende Nachbearbeitung eingesetzt werden. Wie für *Stripey Zebra* führt dies also zur Abwertung.

GERTWOL kombiniert in der Ausgabe der Analyse einer Wortform das Lemma und die Segmentierung in einem String. Anders als für *Morphisto*, ist es jedoch möglich, durch Unterdrücken der Segmentierungszeichen das tatsächliche Lemma zu erhalten. Die C-API kann verwendet werden, um die Ausgabe der Segmentierungszeichen direkt zu verhindern – hier ist also keine Nachbearbeitung notwendig.

Bei der genaueren Betrachtung der Verben fällt eine Fehlerquelle auf, die schon in Abschnitt 6.2.4.3 angesprochen wurde: Verbgefüge.

- Die Wortformen «*plumpsen*», «*nummeriert*», «*stimmen*», «*tupfen*», «*zähle*», «*bettete*», «*pflanzt*», «*zeigt*», «*schlüpfen*» und «*konstruierten*» aus unserer Stichprobe werden als Verbgefüge *und* als Simplex analysiert (d. h., es gibt jeweils eine Analyse mit und eine ohne die Angabe `TRENNBAR`),
- «*durchfeiern*» und «*umstellen*» werden korrekt als Verbgefüge *und* als Kompositum analysiert,
- «*zurechtfänden*», «*übriggeblieben*», «*abstürzt*», «*ausfinanziert*», «*angeknipst*» und «*abzufinden*» werden korrekt ausschliesslich als Verbgefüge analysiert (d. h. mit der Angabe `TRENNBAR`).

Für den ersten Fall wurden diese Analysen mit 3 (alle korrekten, aber auch falsche) bewertet. Die Analysen mit der Angabe `TRENNBAR` sind jeweils falsch. Auch hier hat die neuere *GERTWOL*-Version Verschlechterungen gebracht.

Listing 6.15: GERTWOL *2010: Analyse von* «zähle».

```
zähle
"<zähle>"
        "zähl~en"       V TRENNBAR IND PRÄS SG1
        "zähl~en"       V TRENNBAR KONJ PRÄS SG1
        "zähl~en"       V TRENNBAR KONJ PRÄS SG3
        "zähl~en"       V IND PRÄS SG1
        "zähl~en"       V KONJ PRÄS SG1
        "zähl~en"       V KONJ PRÄS SG3
        "zähl~en"       V IMP PRÄS SG2
(
("zähl~en" . [V PRES IND  SG1])
("zähl~en" . [V PRES CNJV SG1])
("zähl~en" . [V PRES CNJV SG3])
("zähl~en" . [V PRES IND  SG1])
("zähl~en" . [V PRES CNJV SG1])
("zähl~en" . [V PRES CNJV SG3])
("zähl~en" . [V PRES IMP  SG2])
)
```

Listing 6.15 zeigt die Analysen für «*zähle*» mit dem aktuellen *GERTWOL*. Erst durch Betrachtung der *LSLING*-Angaben wird klar, warum in den *LSINDEX*-Angaben scheinbar die gleichen Analysen doppelt ausgegeben werden: Die Angabe `TRENNBAR` wird unterdrückt und damit ist eine solche Analyse identisch mit einer, die diesen Wert ursprünglich nicht hatte.

Listing 6.16: GERTWOL *2000: Analyse von* «zähle».

```
zähle
"<zähle>"
        "zähl~en"  V IND PRÄS SG1
        "zähl~en"  V KONJ PRÄS SG1
        "zähl~en"  V KONJ PRÄS SG3
        "zähl~en"  V IMP PRÄS SG2
```

Vergleichen wir dazu die Analyse mit dem älteren System, erhalten wir nur die Analysen *ohne* die Angabe `TRENNBAR`. Diese Analyse, wie in Listing 6.16 abgebildet, ist vollständig korrekt. In der neueren Version ist also ein Fehler enthalten.

Entsprechend der Beschreibung des Systems von Koskenniemi und Haapalainen [1996] sind für Verben, die einen Verbzusatz erhalten können, zwei Lexikoneinträge vorgesehen, wobei derjenige mit der Markierung `V(sep)` nur verwendet wird, wenn tatsächlich ein Verbzusatz vorhanden ist und das Verbgefüge zusammengeschrieben wird (im Infinitiv, Partizip I und Partizip II). Der Eintrag ohne diese Markierung enthält dagegen eine Markierung `i`, die explizit

macht, dass hier kein Verbzusatz vorhanden ist [Koskenniemi und Haapalainen 1996, S. 124f]. Dass in der neuen Version beide Lexikoneinträge benutzt werden, ist offensichtlich ein Fehler und führt zu einer höheren Anzahl von fehlerhaften Analysen.[39]

Im Gegensatz zu unseren Experimenten in [Mahlow und Piotrowski 2009c] (siehe Abbildung 6.7) liegen die Werte für vollständig korrekte Analysen für *GERTWOL* und *Stripey Zebra* weiter auseinander, die Werte für *GERTWOL* sind für alle Experimente jedoch ähnlich gut. Ein Grund dafür könnte in der Zusammensetzung unserer Stichprobe liegen, die sehr viele Eigennamen (vor allem Personennamen und geographische Bezeichnungen) enthält.[40] Offenbar enthält das Lexikon von *GERTWOL* mehr Eigennamen als das von *Stripey Zebra*.

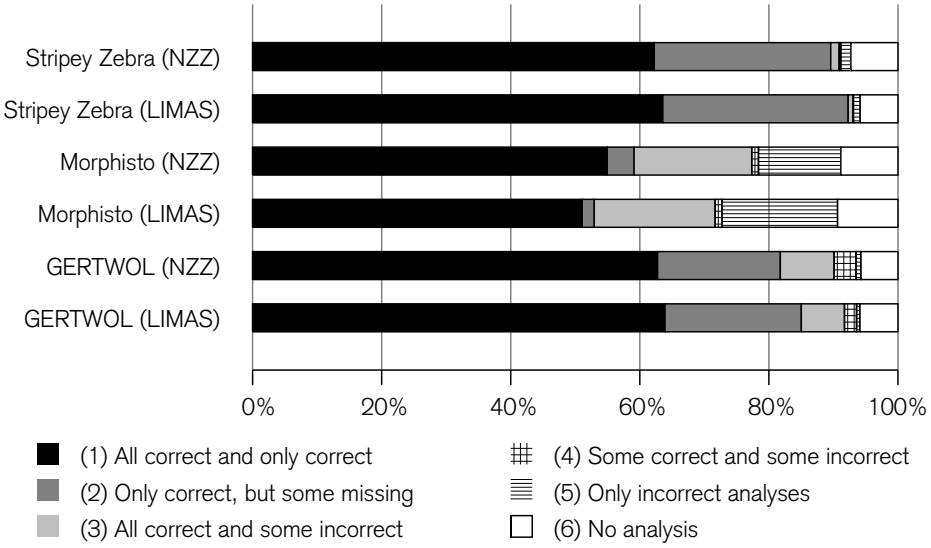


Abbildung 6.7: Detaillierte Auswertung der morphologischen Analysen für eine Stichprobe des LIMAS-Korpus und der NZZ vom April 1994, entnommen aus [Mahlow und Piotrowski 2009c].

6.2.6.4 Zusammenfassung

Die Qualität der Analysen für Adjektive von *Stripey Zebra* und *GERTWOL* könnte mit geringem Aufwand verbessert werden – wenn die Systeme frei zugänglich wären. Da wir für beide Systeme nicht über den Quellcode verfügen, müssten wir die Entwickler kontaktieren und um eine entsprechende Anpassung bitten – da beide Systeme für einen bestimmten Einsatz entwickelt wurden, ist fraglich, ob unserer Anfrage entsprochen würde – oder die Resultate von Analysen nachbearbeiten. Eine solche Nachbearbeitung würde bewirken, dass fast alle Resultate, die jetzt mit 2 bewertet wurden, vollständig korrekt wären. Für *Stripey Zebra* würde sich damit eine Quote von etwa 74 % vollständig korrekten Ergebnissen ergeben, *GERTWOL* würde etwa 71 % vollständig korrekte Resultate liefern. Eine Nachbearbeitung ist jeweils über ein kleines Perl-Skript möglich, da es sich lediglich um nicht explizit dargestellte Informationen handelt, die aus vorhandenen Informationen abgeleitet werden können. Berücksichtigt man die Ergebnisse der Experimente mit anderen Texten (siehe

39. Die Tatsache, dass Verben wie «*kommen*» ausschliesslich als Simplex analysiert werden, weist auf weitere Fehler im Lexikon hin. Derivation, Komposition wie auch die Bildung von Verbgefügen sind produktiv und nicht beschränkt. Wenn zur Analyse von Verbgefügen je Verb zwei Einträge existieren, dann müsste dies für jedes Verb der Fall sein.

40. Da es sich jedoch um eine zufällig ausgewählte Stichprobe handelt und mehrfaches Ziehen einer Stichprobe jeweils ähnliche Eigenschaften aufwies, ist die Stichprobe als repräsentativ anzusehen und die Eigennamen dürfen nicht entfernt oder reduziert werden.

[Mahlow und Piotrowski 2009c], dort werden so 90 % korrekte Ergebnisse für *Stripey Zebra* und 82 % korrekte Ergebnisse für *GERTWOL* ermittelt), können wir festhalten, dass in über 70 % der Fälle, in denen eines der Systeme ein Ergebnis liefert, dieses auch korrekt ist. Beide Systeme sind für den Einsatz in einer Anwendung, die auf korrekter morphologischer Analyse aufbaut, geeignet.[41] Allerdings reichen beide Systeme an eine Quote von «mindestens 90 %» nicht heran, wie sie für die Verwendung in interaktiven Programmen als notwendig erachtet wird, die auf exakte Analyse angewiesen sind (siehe z. B. [Meurers et al. 2010]). Auch die von Hull et al. [1987] geforderte *vollständige* Vermeidung falscher Analysen ist nicht möglich.

Die Mängel von *Morphisto* können nicht durch einfache Nachbearbeitung der Resultate behoben werden. Hier ist intensive Entwicklungsarbeit notwendig (auch um andere Fehler zu beheben), die geleistet werden kann, da das System frei zugänglich ist. Jedoch ist es im Rahmen dieser Arbeit nicht möglich, diese Arbeiten durchzuführen. Ob an *Morphisto* weitergearbeitet wird, ist im Moment nicht absehbar – die ursprünglichen Entwickler haben neue Aufgaben übernommen, die Open-Source-Community müsste sich des Problems annehmen. Vermutlich müssten auch direkt in *SMOR* Anpassungen vorgenommen werden.

Nachdem wir wegen schlechter Abdeckung bereits *mOLIFde* ausgeschlossen haben, müssen wir hier wegen schlechter Qualität der Analysen *Morphisto* ausschliessen. In Frage kommen also nur noch *Stripey Zebra* und *GERTWOL*.

6.2.7 Generierung

Aufgrund der Resultate für die Experimente bezüglich Abdeckung und Qualität der Analysen von Wortformen müssen wir lediglich *Stripey Zebra* und *GERGEN* hinsichtlich der Fähigkeit zur Generierung von Paradigmen und einzelnen Wortformen untersuchen. Die Funktionen, für die wir eine morphologische Komponente als Ressource verwenden wollen, sollen jedoch so implementiert werden, dass die morphologische Ressource leicht gegen eine andere ausgetauscht werden kann. Daraus ergibt sich auch, dass es möglich ist, für die Analyse und die Generierung zwei verschiedene Systeme zu verwenden. Darum betrachten wir hinsichtlich der Generierung noch einmal alle vier Systeme.

6.2.7.1 Stripey Zebra

Das Framework *Malaga* erlaubt es nicht, Wortformen oder Paradigmen zu generieren, es ist ausschliesslich auf morphologische und syntaktische *Analyse* ausgerichtet. Da *Malaga* frei zugänglich ist, könnte es im Hinblick auf Generierung weiterentwickelt werden, dies ist bislang jedoch nicht geschehen und nicht im Fokus unseres Projektes.

Es bietet sich ein Umweg an: *Malaga* erlaubt es, alle gültigen – d. h. analysierbaren – Wortformen zu bilden, die sich aus gegebenen Allomorphen konkatenieren lassen. Daher ist es möglich, aus (a) den Allomorphen einer flektierbaren Grundform und (b) den Flexionsallomorphen alle gültigen Wortformen zu bilden. Werden diese anschliessend analysiert und nur diejenigen Wortformen

41. Die offensichtlichen Fehler in *GERTWOL*, die in früheren Versionen nicht auftraten, sollten auf jeden Fall korrigiert werden.

akzeptiert, die die gleiche Grundform haben wie das Wort, dessen Paradigma gesucht ist, lässt sich so das exhaustive Paradigma dieses Wortes ermitteln. Eine Umwandlung in ein distinktives Paradigma ist möglich. Über diesen Umweg ist es jedoch nicht möglich, eine konkrete Wortform zu generieren. Ebenso muss die benötigte Zeit zur Bildung aller formal korrekten Wortformen und zur anschliessenden Analyse und Auswahl der zum Wort gehörigen Wortformen einberechnet werden.[42] *Stripey Zebra* ist daher für die Generierung nicht geeignet.

6.2.7.2 Morphisto

Die Generierung einer bestimmten Wortform mit *Morphisto* ist als Umkehrung der Analyse implementiert: Für die Analyse einer einzelnen Wortform ist die Oberfläche der Wortform die Eingabe und eine Kombination aus segmentierter Wortform, Wortart und grammatikalischen Eigenschaften die Ausgabe. Für die Generierung ist eine Angabe entsprechend einer Analyse einer Wortform die Eingabe – also eine segmentierte Wortform, Wortart und die gewünschte Kategorie – und die Oberfläche der entsprechenden Wortform die Ausgabe. Die in Abschnitt 6.2.2 geforderten Eingabeformate – das Lemma oder eine Wortform eines Wortes für die Generierung von Paradigmen sowie Lemma und die gewünschte Kategorie für eine konkrete Wortform – sind nicht möglich.

Verwenden wir jedes der drei Analyseergebnisse für «*Mütter*» (siehe Listing 6.2 auf Seite 135) als Eingabe für die Generierung, sollten wir jeweils die Oberfläche «*Mütter*» erhalten – für Nominativ, Genitiv und Akkusativ Plural. Tatsächlich liefert die Generierung jeweils zwei Oberflächen, wie in Listing 6.17 am Beispiel der Genitiv Plural-Form zu sehen.

Listing 6.17: Morphisto*: Generierung des Genitiv Plural von* «Mütter».[46]

```
generate> Mutter<+NN><Fem><Gen><Pl>
Mutter<+NN><Fem><Gen><Pl>
Mütter
Muttern
```

Die Ursache liegt darin, dass die Grundform «*Mutter*» ambig ist und verschieden flektiert wird, je nachdem ob es sich um das Gegenstück zur Schraube oder um die Mutter eines Kindes handelt. Da die Flexionsklasse nicht explizit angegeben wird, werden die Oberflächen der entsprechenden Wortformen für beide Wörter zurückgegeben. Die Flexionsklasse ist im Lexikon jeweils für ein Lemma vermerkt – im Lexikon sind tatsächlich Lemmata enthalten, nicht segmentierte Wortformen. Die Ausgabe der Generierungskomponente ist unter diesen Umständen also korrekt.

Für Wörter, die nicht komplex sind, d. h., deren Grundform nicht durch Komposition oder Derivation gebildet wird (wie «*Mutter*»), ist die Oberfläche der segmentierten Wortform mit dem Lemma identisch (siehe Abschnitt 6.2.4.2 auf Seite 135). Für komplexe Wörter ist die Generierung jedoch schwieriger und zudem fehlerhaft. Um den Genitiv Singular von «*Leistungsnachweis*» zu

42. Für Sprachen mit relativ grossem Allomorphiequotient wie Spanisch werden bis zu 8 Sekunden benötigt, um ein vollständiges Paradigma eines Verbs zu generieren, wie in [Mahlow und Piotrowski 2009a] beschrieben.

46. *Morphisto* liefert jeweils nach der Eingabe in der ersten Zeile noch einmal die Eingabe und dann die generierten Wortformen.

erhalten, würde die gewünschte Eingabe entsprechend der *Morphisto*-Syntax wie folgt lauten:

```
Leistungsnachweis<+NN><Masc><Gen><Sg>
```

also eine Kombination aus Wort und gewünschter Kategorie. Diese Eingabe führt jedoch zu keinem Ergebnis. Wir müssen die Analysen von «*Leistungsnachweises*» verwenden (siehe Listing 6.4 auf Seite 136 – beide Analysen enthalten nicht die Grundform «*Leistungsnachweis*») und erhalten drei verschiedene Oberflächen: «*Leistungsnachweises*», «*Leistungennachweises*» und «*Leistungnachweises*», siehe Listing 6.18.

Listing 6.18: Morphisto*: Generierung des Genitiv Plural von* «Leistungsnachweis».

```
Leistung<NN>Nachweis<+NN><Masc><Gen><Sg>
> Leistung<NN>Nachweis<+NN><Masc><Gen><Sg>
Leistungsnachweises
Leistungennachweises
Leistungnachweises

leisten<V>ung<NN><SUFF>Nachweis<+NN><Masc><Gen><Sg>
> leisten<V>ung<NN><SUFF>Nachweis<+NN><Masc><Gen><Sg>
Leistungsnachweises
Leistungnachweises
```

Interessant sind hier zwei Aspekte: (a) Abhängig davon, welche der ursprünglichen Analysen als Eingabe verwendet wird, werden unterschiedliche Wortformen und auch unterschiedlich viele Wortformen generiert. (b) Da die Grundform des Wortes («*Leistungsnachweis*») nicht mit angegeben wird und in der Analyse das Fugenelement nicht explizit aufgeführt wird, steht für die Generierung die Information über zu verwendende Fugenelemente nicht zur Verfügung. Darum werden Wortformen mit prinzipiell möglichen (d. h. formal validen) Fugenelementen erzeugt. Alle generierten Formen können auch analysiert werden (siehe Listing 6.19), sind bis auf «*Leistungsnachweises*» jedoch nicht wohlgeformt. Im Gegensatz zu *GERTWOL*, das nur wohlgeformte Wortformen erkennt, erkennt und analysiert *Morphisto* also auch nichtwohlgeformte Wortformen.

Listing 6.19: Morphisto*: Analyse der Genitiv Plural-Formen von* «Leistungsnachweis».

```
Leistungsnachweises
> Leistungsnachweises
Leistung<NN>Nachweis<+NN><Masc><Gen><Sg>
leisten<V>ung<NN><SUFF>Nachweis<+NN><Masc><Gen><Sg>

Leistungennachweises
> Leistungennachweises
Leis<NN>Tun<NN>Gen<NN>Nachweis<+NN><Masc><Gen><Sg>
Leistung<NN>Nachweis<+NN><Masc><Gen><Sg>
leisten<V>Un<OTHER>Gen<NN>Nachweis<+NN><Masc><Gen><Sg>

Leistungnachweises
> Leistungnachweises
Leistung<NN>Nachweis<+NN><Masc><Gen><Sg>
leisten<V>ung<NN><SUFF>Nachweis<+NN><Masc><Gen><Sg>
```

Für die Generierung von Verbgefügen wie «*hinauftragen*» entspricht die Ausgabe von *Morphisto* nicht unseren Anforderungen. Für die Eingabe:

```
hinauf<PREF>tragen<+V><3><Sg><Pres><Ind>
```

(d. h., gesucht ist die Wortform für die 3. Person Singular Präsenz Indikativ von «*hinauftragen*») wird als Resultat «*hinaufträgt*» zurückgeliefert. Dies ist jedoch

nur eine der gesuchten Formen zur Verwendung in Nebensätzen, die Form «*trägt hinauf*» zur Verwendung in Hauptsätzen wird nicht erzeugt.

Das Paradigma eines Wortes wird über einen Umweg generiert. Hierfür wird als Eingabe die Grundform des Wortes verwendet. Erzeugt werden exhaustive Paradigmen, indem für jede Kategorie eines Paradigmas die gesuchten Wortformen einzeln erzeugt werden. Die Probleme mit Verbzusätzen für Verbgefüge gelten natürlich auch für die Erzeugung von Paradigmen. Für «*Leistungsnachweis*» wird das korrekte Paradigma erzeugt. Da es jedoch nicht möglich ist, für eine komplexe Wortform das Lemma korrekt zu bestimmen, kann – ausgehend von einer konkreten Wortform – das gesamte Paradigma des Wortes nicht ermittelt werden.

Morphisto ist generell für die Generierung geeignet, jedoch nicht für das Generieren von Paradigmen. Die schon geschilderten Mängel hinsichtlich der Lemmatisierung wirken sich auch auf die Generierung aus und machen es unmöglich, *Morphisto* als Generierungskomponente einzusetzen.

6.2.7.3 GERGEN

Da es sich bei *GERTWOL* um einen Transducer ähnlich wie *SMOR* (also *Morphisto*) handelt, ist der invertierte Transducer für die Generierung geeignet. Diese Version ist *GERGEN*. Die Eingabe zur Generierung einer Wortform ist die Grundform und die gewünschte Kategorie entsprechend der Ausgabe für die Analyse einer Wortform. Für die Eingabe:

```
Mutter+S FEM DAT PL
```

erhält man die Formen «*Müttern*» und «*Muttern*». Beide Formen sind korrekt, wie im Abschnitt über *Morphisto* bereits angemerkt. Die Eingabe entspricht weitestgehend unseren Anforderungen an eine Generierungskomponente, die Ergebnisse sind auf den ersten Blick zufriedenstellend. Es ist jedoch nicht möglich, das Paradigma eines Wortes zu generieren, auch wenn diese Möglichkeit in der Dokumentation genannt wird.[47] Um das gesamte Paradigma eines Wortes zu erhalten, müssen also wie für *Morphisto* alle einzelnen Formen explizit generiert werden.

Für ein Verbgefüge wie «*hinauftragen*» wird jedoch nur die zusammengeschriebene Form zurückgeliefert. Für die Verwendung solcher Verbformen in Hauptsätzen muss die Ausgabe also nachbearbeitet werden – d. h., die Tatsache, dass es sich um ein Verbgefüge handelt, muss bekannt sein. Zudem muss für Verbgefüge die Angabe `TRENNBAR` der Generierungskomponente übergeben werden.

Dass ein Verb ein Verbgefüge, eine Derivation, ein Kompositum oder ein Simplex ist, ist zwar eine Eigenschaft des Wortes, gehört jedoch nicht zur Kategorie einer bestimmten Wortform. Vergleichbar ist diese Information mit der Flexionsklasse eines Wortes. Beide Angaben gehören nicht zu den von uns in Abschnitt 6.2.2 definierten Eingabeformaten für die Generierung einer bestimmten Wortform oder eines Paradigmas. Diese Information kann dagegen

47. Die Auskunft, dass diese Angabe in der Dokumentation tatsächlich falsch ist, stammt von einem der Entwickler bei der Firma Lingsoft, Marko Nyman, E-Mail vom 8. Mai 2009, 12:05, Message-ID `<4A03F5E4.4040405@lingsoft.fi>`. Die Funktionalität wurde bislang nicht nachgebessert.

aus dem Lexikoneintrag für das entsprechende Wort ermittelt – wie es ja auch für die Analyse geschieht – und dann zur Generierung verwendet werden. Hier zeigt sich, dass das simple «Umkehren» eines Transducers, der für die Analyse wertvolle zusätzliche Informationen neben Lemma und Kategorie liefert, nicht der optimale Weg für die Generierung ist.

Mit den zusätzlich notwendigen Angaben stossen wir auf ein weiteres Designproblem des aktuellen *GERTWOL/GERGEN*: Unter Verwendung von *LSLING* ist für «*geh*» die Information `GESPROCHEN` neben der Kategorie `V IMP PRÄS SG2` angegeben. Diese Information wird in der Ausgabe unter Verwendung von *LSINDEX* unterdrückt, dort erhalten wir nur die Kategorie `V PRES IMP SG2`. Ebenso verhält es sich mit Angaben wie `SELTEN` für veraltete Formen des Dativs für Substantive, Angaben zur starken oder schwachen Deklination für Adjektive oder `GESPROCHEN` für Formen wie «*gibt's*» (hier wird zusätzlich noch die Grundform des Pronomens mitgeliefert). Wortformen wie «*im*», also mit einem Pronomen verschmolzene Artikel («*in dem*»), erhalten mit *LSLING* eine Analyse, die zwei Wortarten (`PRÄP` und `DET`) angibt – wird diese Ausgabe auf *LSINDEX* abgebildet, das nur die Angabe *eines* Wertes für die Wortart erlaubt, wird eine Angabe (hier `PRÄP`) unterdrückt.

Die ursprüngliche Annahme, dass ähnlich der Abbildung der Analyseinformationen aus *LSLING-* in *LSINDEX-*Format für die Generierung eine Abbildung aus *LSINDEX-* in *LSLING-*Format erfolgt, hat sich nicht bestätigt. Die naheliegendere Vermutung, dass direkt unter Verwendung der *LSINDEX-*Angaben die Generierung erfolgt, wurde von den Entwicklern verneint, es wird intern mit den *LSLING-*Angaben gearbeitet.[48] Es ist also notwendig, für die Generierung die Angaben der *LSLING-*Analysen zu verwenden. Eine Verwendung der *LSINDEX-*Angaben mit einer Umwandlung der kategorialen Bezeichnungen in die entsprechenden *LSLING-*Terme reicht nicht aus, da für etliche Formen die nicht zur Kategorie gehörenden, aber von der Analyse ermittelten Zusatzangaben notwendig sind. So muss für Verbgefüge die Information `TRENNBAR` mit übergeben werden, für Formen wie «*gibt's*» die Information `GESPROCHEN` und das Pronomen «*es*».

Interessant ist zudem, dass einige Wortformen, wie «*hat's*» (also eine ähnliche Form wie «*gibt's*»), analysiert, jedoch nicht generiert werden können. Die Generierung ist offensichtlich also nicht eine simple «Umkehrung» der Analyse. Wir müssen ebenfalls festhalten, dass die vielversprechende Form der Analyseergebnisse als Liste geordneter Tags (*LSINDEX*) nicht zu verwenden ist, da wir für die Generierung direkt mit den Ergebnissen als unformatiertem String arbeiten müssen. Dieser Fakt ist für Funktionen relevant, die sehr abstrakt darauf basieren, dass für eine beliebige Wortform die Kategorie ermittelt, ein Wert (etwa Tempus für ein Verb oder Numerus für ein Substantiv) verändert und die entsprechende Wortform zurückgeliefert werden sollen.

6.2.7.4 mOLIFde

mOLIFde kann sowohl für die Analyse von Wortformen als auch zur Generierung von Paradigmen eingesetzt werden. Zur Generierung werden Angaben benötigt zu: (a) OLIF-Flexions-Code, (b) Lemma, (c) Wortart und (d) Ausgabe-

48. Auskunft von Marko Nyman, E-Mail vom 13. Juli 2010, 17:34, Message-ID `<4C3C7987.80504@lingsoft.fi>`.

format. Für das Ausgabeformat gibt es drei Varianten: (a) EAGLES, (b) STTS (Stuttgart-Tübingen-TagSet) und (c) GB (grammarbook). Die Ausgabe ist ein exhaustives Paradigma. Wortart und Lemma lassen sich aus der Analyse des Wortes ermitteln, ebenso der OLIF-Flexions-Code (siehe Listing 6.9 auf Seite 139). Damit können die Analysedaten einer Wortform verwendet werden, um das gesamte Paradigma dieses Wortes generieren zu lassen. Das Eingabeformat entspricht weitestgehend unseren Anforderungen. Allerdings muss entweder der OLIF-Flexions-Code jeweils bekannt sein oder zuerst durch die Analyse der Grundform ermittelt werden. Wie für *GERGEN* (hier zur Ermittlung der Angaben wie `TRENNBAR` oder `GESPROCHEN`) muss erst die Analysekomponente aufgerufen werden, was den Generierungsprozess insgesamt verlängert. Wie *Morphisto* und *GERGEN* kann auch *mOLIFde* Paradigmen für Verbgefüge nicht vollständig korrekt bilden, es wird jeweils nur die Wortform zur Verwendung in Nebensätzen ausgegeben.

6.2.7.5 Zusammenfassung

In Abschnitt 6.2.6.4 auf Seite 153 haben wir festgehalten, dass aufgrund der Eigenschaften der Systeme bezüglich morphologischer Analyse von Wortformen nur *GERTWOL/GERGEN* und *Stripey Zebra* ernsthaft zu berücksichtigen seien. Für die Generierung haben wir dann alle vier Systeme betrachtet.

Die Probleme, die *mOLIFde* und *Morphisto* im Bereich der Analyse haben – schlechte Abdeckung, keine Ermittlung des Lemmas –, setzen sich in der Generierung fort. *Stripey Zebra* ist prinzipiell nicht für die Generierung geeignet, ein «workaround» benötigt für die Generierung so viel Zeit, dass sie für den interaktiven Einsatz nicht in Frage kommt. *GERGEN* weist in der uns zur Verfügung stehenden aktuellen Version erstaunliche Mängel auf.

Unter Berücksichtigung der Eigenschaften der anderen Systeme erachten wir *GERGEN* gleichwohl als am besten geeignetes System, aus der Eingabe von Lemma, Wortart und gewünschter Kategorie die entsprechende Wortformoberfläche zu generieren. In der Verwendung der Ergebnisse müssen die Mängel jedoch entsprechend berücksichtigt werden.

6.2.8 Zusammenfassung: Entscheidung für eine morphologische Ressource

In diesem Abschnitt haben wir vier uns zur Verfügung stehende regelbasierte Systeme zur Analyse und Generierung von deutschen Wortformen untersucht. Wir verglichen *Stripey Zebra*, *Morphisto*, *GERTWOL/GERGEN* und *mOLIFde* hinsichtlich Verfügbarkeit, Schnittstellen, Format der Resultate und Möglichkeiten zu deren Weiterverarbeitung, Abdeckung und Geschwindigkeit, Qualität der Analysen sowie Fähigkeit zur Generierung. Dazu haben wir verschiedene Experimente durchgeführt.

Wir möchten hier noch einmal darauf hinweisen, dass unsere Evaluation keine generelle Aussage über die Qualität der betrachteten Systeme liefert. Vielmehr wurden in Abschnitt 6.1 und Abschnitt 6.2.2 spezifische Anforderungen definiert: Ziel der Evaluation war es, das am besten geeignete morphologische System zu finden, das für die Einbindung in interaktive Funktionen in einem Editor geeignet ist. Die Ergebnisse der Geschwindigkeits- und Abdeckungstests

und der Untersuchung der Analysen geben einen allgemeinen Eindruck bezüglich der Qualität der einzelnen Systeme, für eine valide allgemeine Aussage wäre jedoch eine Evaluation gegen einen Gold-Standard notwendig. Sieht man von diesen Einschränkungen ab, ist dies die erste umfangreiche Evaluation von morphologischen Systemen für Deutsch seit der ersten Morpholympics 1994.

Tabelle 6.4 fasst die Ergebnisse zusammen. Für *mOLIFde* wurde die Qualität der Analysen nicht untersucht, da die Abdeckungstests sehr negativ ausfielen.

*Tabelle 6.4: Zusammenfassung der Resultate der Evaluation. Mit * markierte Kriterien sind entscheidend, die anderen können vernachlässigt werden, beeinflussen jedoch die endgültige Entscheidung.*

Kriterium	***Stripey Zebra***	***Morphisto***	***GERTWOL/ GERGEN***	***mOLIFde***
Open Source	±	+	−	±
Einfach zu installieren	+	−	+	+
*Schnittstellen	+	±	+	±
*Weiterverarbeitung der Resultate	+	±	+	±
Geschwindigkeit	+	+	+	+
*Abdeckung	+	+	+	−
*Qualität der Analysen	+	±	+	n/a
*Generierung	−	±	±	±

Berücksichtigt man alle relevanten Eigenschaften der Systeme und die vorab postulierten Bedingungen, ist nur *GERTWOL/GERGEN* als geeignet für die Einbindung in interaktive Funktionen zur Unterstützung des Redigierens einzustufen:

- *Stripey Zebra* benötigt wenige Ressourcen zum Kompilieren und während der Laufzeit, die Abdeckung ist mit etwa 95 % sehr gut und die Geschwindigkeit zufriedenstellend. Es hat von den getesteten Systemen das grösste Lexikon und liefert die detailliertesten Angaben zur Analyse einer Wortform. Die Qualität dieser Analysen ist ebenfalls sehr gut. Aufgrund unserer Experimente können wir, abhängig von den analysierten Texten, erwarten, 60 bis 90 % vollständig korrekte Ergebnisse zu erhalten. Da *Stripey Zebra* die morphologischen Prinzipien der Derivation und Komposition nachbildet, ist es nicht notwendig, abgeleitete Wörter und Komposita im Lexikon aufzuführen. Auch ad hoc erzeugte neue Wörter wie komplexe Komposita werden ohne Probleme erkannt. Mit *Perl-Malaga*, *Python-Malaga*, und *Ruby-Malaga* existieren drei Schnittstellen, die eine einfache Weiterverarbeitung der Analyseergebnisse und die Einbindung in andere Applikationen ermöglichen. *Stripey Zebra* kann sowohl Wortformen nach neuer als auch nach alter Rechtschreibung analysieren, ebenso ist Schweizer Rechtschreibung (d. h. keine Verwendung von «ß», sondern nur «ss») möglich. Jedoch können keine Wortformen oder gar Paradigmen generiert werden, zudem ist es nicht frei verfügbar. *Stripey Zebra* kann also nicht in Applikationen integriert werden, die über eine Open Source Lizenz zur Verfügung gestellt werden sollen.

- *Morphisto* ist frei zugänglich, das System kann bei Bedarf selbst angepasst werden, die Geschwindigkeit und auch die Abdeckung sind sehr gut, obwohl es das kleinste Lexikon hat. Es sind Schnittstellen zur Einbindung in andere Applikationen vorhanden. Ein Nachteil sind die sehr

hohen Hardware-Anforderungen zur Kompilierung des Lexikons und die dafür benötigte Zeit. Anschliessend wird für Analyse und Generierung nicht viel Rechenleistung benötigt, die Analyse ist extrem schnell, liefert teilweise jedoch sehr viele Analysen je Wortform. Es wird kein Lemma für eine analysierte Wortform geliefert, *Morphisto* erfüllt die grundlegenden Anforderungen an eine morphologische Analysekomponente damit nicht. Die Generierung ist sowohl für konkrete Wortformen als auch für Paradigmen von Wörtern möglich, die Qualität ist jedoch nicht zufriedenstellend.

- *GERTWOL* hat eine leicht bessere Abdeckung als *Stripey Zebra* und vergleichbare Werte hinsichtlich Geschwindigkeit und Qualität der Analysen. Unsere Experimente zur Qualität der Analysen haben gezeigt, dass, abhängig von den analysierten Texten, vollständig korrekte Ergebnisse für 65 bis 85 % der Wortformen zu erwarten sind. Eine Erweiterung des Lexikons oder der Regeln ist wie für *Stripey Zebra* nicht möglich. *GERGEN* ist die zugehörige Generierungskomponente, die die Generierung konkreter Wortformen, jedoch nicht von Paradigmen, ermöglicht. Die Qualität von *GERGEN* ist nicht vollständig zufriedenstellend, jedoch entspricht das Eingabeformat bzw. die benötigten Informationen am ehesten unseren Anforderungen. Aufgrund des Lizenzsystems ist es nicht möglich, *GERTWOL/GERGEN* in Applikationen einzubinden, die frei zur Verfügung gestellt werden sollen.

- *mOLIFde* ist zwar frei zugänglich, die Abdeckung ist mit etwa 30 % jedoch extrem schlecht, da nur einige Wortarten behandelt werden. Dies ist umso erstaunlicher, als *mOLIFde* ein sehr grosses Lexikon hat. *mOLIFde* kann nur die Wortformen erkennen, deren Lemmata im Lexikon vorhanden sind und deren Wortart vorab korrekt erkannt wurde. Neue unbekannte Wörter und vor allem komplexe Komposita aus Elementen, die bereits im Lexikon vorhanden sind, können nicht erkannt werden. Zudem sind die notwendigen grundlegenden Werkzeuge (XFST) nicht frei zugänglich, *mOLIF* kann also nicht in andere Open-Source Applikationen integriert werden. Die Kosten zum Installieren sind gering, die Geschwindigkeit ist sehr hoch. Ein Vorteil von *mOLIFde* ist die Fähigkeit zur Generierung von Paradigmen. Die Generierung einer Wortform mit einer bestimmten Kategorie ist nicht möglich, sondern muss über den Zugriff auf das gesamte Paradigma simuliert werden.

Die konkrete Einbindung von *GERTWOL/GERGEN*in den Editor *XEmacs* und entsprechende linguistisch motivierte Funktionen beschreiben wir in Abschnitt 7.2.1 auf Seite 197 und in den Abschnitten 7.3.4 und 7.3.5.

Wir haben gezeigt, dass *GERTWOL/GERGEN* das aktuell am besten geeignete System für morphologische Analyse und Generierung ist. Es ist daher nicht überraschend, dass in Arbeiten, die morphologische Analyse verwenden, *GERTWOL* als Ressource verwendet wird. Wir konnten jedoch auch zeigen, dass die aktuelle Version gegenüber einer älteren Version von 2000 erhebliche Mängel aufweist. Es erstaunt daher, dass in Arbeiten, wie von Hardmeier et al. [2010], Hjelm und Schwarz [2005], Klenner et al. [2010a,b], Niessen und Ney [2000] oder Sennrich et al. [2009], die tatsächlich verwendete Version von *GERTWOL* nicht genannt wird. Da in einigen Fällen wie von Hjelm und Schwarz [2005] Beispielanalysen gezeigt werden, kann jedoch vermutet werden,

dass eine alte Version, die etwa unserer Version von 2000 entspricht, verwendet wird – da diese Version nicht mehr erhältlich ist, können die publizierten Ergebnisse vermutlich nicht verifiziert werden. Ebenso überrascht, dass diese Autoren nicht darauf eingehen, welchen Effekt der Einsatz von *GERTWOL* hat bzw. wie Qualität und Abdeckung beurteilt werden in Bezug auf den eigentlichen Einsatz im Information Retrieval, zur Anaphernresolution oder in der maschinellen Übersetzung.

6.3 Automatische Wortartenbestimmung

Um zu entscheiden, ob eine Wortform ein Adjektiv oder ein Verb ist, ist keine vollständige morphologische Analyse notwendig, solange die Kategorie der Wortform nicht benötigt wird. Die Zuweisung der Wortart mittels automatischer Wortartenbestimmung (engl. *part-of-speech-tagging*) ist dafür ausreichend. Im Gegensatz zur morphologischen Analyse, die jeweils nur die zu analysierende Wortform behandelt, wird für die Zuweisung der Wortart auch der Kontext einer Wortform, d. h. der Satz, berücksichtigt. Es wird jeweils genau eine Wortart für jede Wortform zurückgeliefert – hier findet also Gewichtung statt. Entsprechende Programme können regelbasiert oder statistisch arbeiten, benötigen jedoch in jedem Fall ein annotiertes Trainingskorpus, siehe [Schmid 2008].

Wir wählen zunächst in Abschnitt 6.3.1 ein Programm aus, das unseren allgemeinen Anforderungen in Abschnitt 6.1 entspricht, und begründen anschliessend in Abschnitt 6.3.2, welches Trainingskorpus wir verwenden.

6.3.1 Auswahl der Software zur Bestimmung von Wortarten

Seit mehreren Jahren ist der *TreeTagger* [Schmid 1994, 1995], entwickelt von Helmut Schmid am Institut für Computerlinguistik der Universität Stuttgart, die Standard-Software, wenn es um die Wortartenbestimmung als Teil umfassender Anwendungen geht, wie etwa in der Bestimmung der Thematik von Texten [Ferret et al. 1998, Hernandez und Grau 2003], im Dokumentenretrieval [Hollink et al. 2004], zur Bestimmung der Qualität von Diskussionsbeiträgen [Weimer et al. 2007], in der statistischen maschinellen Übersetzung [Niehues und Kolss 2009] oder zur Ermittlung von Mehrwortausdrücken [Lieberman et al. 2010]. Für *TreeTagger* wird jeweils Korrektheit von über 95 % angegeben. *TreeTagger* ist für wissenschaftliche Zwecke frei verfügbar, jedoch erhält man den Quelltext nicht. Es ist möglich, *TreeTagger* selbst auf einem entsprechenden Korpus zu trainieren oder bereits erstellte Parameterdateien zu verwenden.

Auf den ersten Blick ist die Situation also sehr viel komfortabler als im Bereich morphologischer Analyse: Es existiert ein frei verfügbares System, das eine sehr hohe Qualität aufweist und weiter aktiv gepflegt wird. Jedoch war bei der Entwicklung von *TreeTagger* Anfang der 1990er Jahre nicht der interaktive Einsatz im Fokus und alle Projekte, die ihn einsetzen, sind ebenfalls nicht auf interaktive Benutzung ausgelegt. *TreeTagger* lässt sich zwar relativ schnell starten, aber nicht schnell genug, um ihn bei Bedarf für jeden Satz oder jede Phrase neu zu starten. Es wäre also notwendig, den Prozess nach der Analyse einer Eingabe nicht zu beenden, sondern den Prozess im Hintergrund laufen zu lassen und

ihm jeweils eine Eingabe zu schicken. Für die Lösung dieses Problems existieren verschiedene *Wrapper*, wie etwa das Python-Modul *treetaggerwrapper*[49].

Es kommt jedoch noch ein weiteres Problem hinzu: Das System verwendet gepufferte Ein-/Ausgabe. Die Verarbeitung einer Eingabe und die Ausgabe des Resultats erfolgen also erst, wenn der Eingabepuffer voll ist – d. h., die erste eingegebene Wortform wird nicht direkt verarbeitet, sondern erst nach der Eingabe weiterer Wortformen. Es gibt keine Möglichkeit, die sofortige Verarbeitung und Ausgabe des Ergebnisses zu erzwingen. *TreeTagger* ist damit für die nicht-interaktive Bearbeitung grosser Textmengen sehr gut geeignet, nicht jedoch für den interaktiven Einsatz.

Listing 6.20: Aufruf und Verwendung von TreeTagger.

```
$ tree-tagger /opt/treetagger/lib/german.par -token -lemma
        reading parameters ...
        tagging ...
Auf
den
ersten
Blick
>>      Auf     APPR    auf
ist
>>      den     ART     d
die
>>      ersten  ADJA    erst
Situation
>>      Blick   NN      Blick
komfortabel
>>      ist     VAFIN   sein
.
>>      die     ART     d
^D
>>      Situation       NN      Situation
>>      komfortabel     ADJD    komfortabel
>>      .       $.      .
>>              finished.
```

Listing 6.20 zeigt den Aufruf von *TreeTagger* in der Konsole und die Eingabe des Satzes «*Auf den ersten Blick ist die Situation komfortabel.*» sowie die Ausgabe. Die Verarbeitung der ersten Wortform («*Auf*») und die Ausgabe des Ergebnisses beginnt erst nach Eingabe der vierten Wortform («*Blick*»). Die Ausgaben des Systems sind jeweils eingerückt und mit `>>` gekennzeichnet. Nach Eingabe des abschliessenden Satzzeichens ist die Verarbeitung noch nicht beendet, die Verarbeitung und Ausgabe der noch fehlenden Wortformen wird durch Schliessen der Eingabe und damit Abbruch des Prozesses (`^D`) erzwungen. Das bereits erwähnte Python-Modul führt zur Umgehung dieses Problems einen «Dummy-Satz» ein, der jeweils nach dem letzten Wort der eigentlichen Eingabe an *TreeTagger* geschickt wird und die vollständige Analyse und Ausgabe der Ergebnisse auslöst.[50]

Als Alternative bietet sich die Verwendung des Wortartenbestimmers *Mbt* [Daelemans et al. 1996, 2010, Zavrel und Daelemans 1999] an, entwickelt von der Forschungsgruppe «Induction of Linguistic Knowledge» am Department of Communication and Information Sciences der Universität Tilburg. *Mbt* steht für *memory-based tagger*. Das Framework *Mbt* dient dazu, einen Wortartenbestimmer zu generieren und diese Komponente dann auszuführen, es basiert

49. http://www.limsi.fr/Individu/pointal/python/treetaggerwrapper-doc/treetaggerwrapper-module.html (zuletzt besucht am 8.12.2010, 19:34).
50. Siehe http://www.limsi.fr/Individu/pointal/docs/doku.php?id=dev:treetaggerwrapper (zuletzt besucht am 8.12.2010, 19:34).

auf *TiMBL* (*Tilburg Memory-Based Learner*) [Daelemans und van den Bosch 2005]. Beide Komponenten sind frei erhältlich unter der GNU General Public License.

Die Verwendung von *Mbt* ist nicht so stark vertreten wie die von *TreeTagger*, doch lässt sich für *Mbt* ebenfalls der Einsatz in verschiedensten Anwendungen belegen: flache syntaktische Analyse [van den Bosch und Buchholz 2002], Bestimmung von Wortbedeutungen [Hoste et al. 2002], linguistische Untersuchungen [Ivanova und Kübler 2008] oder Korrektur von Annotationen in Korpora [Loftsson 2009].

Mbt verwendet ein annotiertes Trainingskorpus, um daraus ein Lexikon für die enthaltenen Wortformen und Heuristiken für unbekannte Wortformen zu generieren. Diese Informationen werden dann für die automatische Zuweisung von Wortarten für Wortformen in anderen Texten verwendet. Die Qualität eines *Mbt*-Taggers hängt von der Grösse und Qualität des Trainingskorpus ab, wie es auch für *TreeTagger* gilt. Die jeweiligen Autoren berichten ebenfalls von Korrektheit von über 95 %.

Im Gegensatz zu *TreeTagger* lässt sich *Mbt* problemlos in interaktiven Applikationen einsetzen. Die Einbindung in den Editor *XEmacs* kann mit wenigen Zeilen ELisp-Code erfolgen. Die Ergebnisse der Verarbeitung einer Eingabe werden jeweils sofort zurückgeliefert, es ist nicht notwendig, «Dummy-Eingaben» wie für *TreeTagger* zu verwenden. Wir verwenden daher für unsere Funktionen *Mbt* und stellen im folgenden Abschnitt 6.3.2 dar, welches Korpus wir für die Generierung des Wortartenbestimmers verwenden.

6.3.2 Auswahl des Trainingskorpus

Die bekanntesten und am häufigsten verwendeten annotierten Korpora für geschriebenes Deutsch sind *NEGRA* [Brants et al. 1997], *TiGer* [Brants et al. 2002] und *TüBa-D/Z* [Telljohann et al. 2009]. Alle sind entsprechend dem *Stuttgart-Tübingen Tagset* (STTS) [Schiller et al. 1999] ausgezeichnet. Alle drei Korpora stehen uns zur Verfügung.

- Das *NEGRA*-Korpus in der aktuellen Version (Version 2) aus dem Jahr 2006 enthält 355'096 laufende Wortformen in 20'602 Sätzen. Es beruht auf Texten der Zeitung «Frankfurter Rundschau». Diese Texte stammen aus dem «Multilingual Corpus I» der European Corpus Initiative aus dem Jahr 1994. Annotiert sind Argument-Strukturen, grammatische Funktionen sowie syntaktische und morphologische Informationen. Das Korpus ist für wissenschaftliche Zwecke frei verfügbar.[51]

- Das *TiGer*-Korpus in der aktuellen Version (Version 2.1) enthält 900'000 laufende Wortformen in ca. 50'000 Sätzen. Es beruht ebenfalls auf Texten der Zeitung «Frankfurter Rundschau». Die Annotation erfolgte semi-automatisch, ausgezeichnet sind syntaktische Strukturen und morphologische Informationen für Wortformen. Für wissenschaftliche

51. Siehe `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html` (zuletzt besucht am 8.12.2010, 19:34).

Zwecke ist das Korpus frei zugänglich. Das Korpus wird nicht mehr aktiv gepflegt und entwickelt.[52]

- *TüBa-D/Z* (Tübinger Baumbank des Deutschen/Schriftsprache) ist ein syntaktisch manuell annotiertes Korpus aus Texten der Zeitung «die tageszeitung» (taz) aus den Jahren 1992, 1995 und 1999. Die Annotationsarbeit wird fortgesetzt, etwa jährlich erscheint ein neues Release des Korpus, das zusätzliche annotierte Texte enthält (alle aus den ursprünglichen taz-Ausgaben). Version 5 enthält 794'079 laufende Wortformen in ca. 45'200 Sätzen [Telljohann et al. 2009]. Das Korpus ist frei verfügbar, sofern man das elektronische Archiv der «taz» erworben hat.

TüBa-D/Z und *NEGRA* enthalten ausschliesslich Texte entsprechend den Regeln *vor* der Rechtschreibreform. Sie spiegeln also nicht aktuell geschriebene Sprache wider. Alle drei Korpora basieren auf Zeitungstexten – das macht sie untereinander gut vergleichbar, deckt jedoch nur ein Genre ab.

Unseres Wissens existiert nur ein annotiertes Korpus mit Texten verschiedener Genres in *aktueller* Rechtschreibung: Die deutschen Texte in der *SMULTRON*-Treebank [Gustafson-Čapková et al. 2007][53]. Dieses Korpus ist jedoch mit insgesamt 1'047 Sätzen sehr klein[54] und enthält keine morphologischen Informationen – als morphologische Information wird lediglich für Substantive das Genus ausgezeichnet.

Die Situation in Bezug auf annotierte Korpora als Trainingsgrundlage für statistische computerlinguistische Komponenten ist also eher unbefriedigend: Wir sind gezwungen, Daten zu verwenden, die nicht für aktuelle Texte entsprechend den aktuellen Orthographieregeln geeignet sind. Beispielsweise werden einige Vorkommen von «*dass*» (subordinierende Konjunktion) durch den Wortartenbestimmer fälschlich als `Artikel` markiert (da «*dass*» nicht in den Trainingsdaten vorkommt, wird hier eine Hypothese geliefert); Verbgefüge werden entsprechend den neuen Regeln häufiger getrennt geschrieben. Interessanterweise wird in aktuellen Publikationen, die über Projekte und Experimente unter Verwendung von annotierten Korpora für das Deutsche berichten, nicht darauf eingegangen, dass diese Korpora für die Bearbeitung aktueller Texte nicht geeignet sind [vgl. Boyd und Meurers 2008, Evert 2004, Hall und Nivre 2008, Kübler et al. 2006, Nicholson et al. 2008, Rafferty und Manning 2008], obwohl bekannt ist, dass die Fehlerrate erheblich steigt, wenn das Trainingskorpus nicht den tatsächlich bearbeiteten Texten entspricht [vgl. Foster 2010, Schmid 2008, S. 547f].

Experimente mit *NEGRA*, *TiGer* und *TüBa-D/Z* als Trainingskorpus zeigen, dass keines der Korpora hinsichtlich der Korrektheit der Wortartenbestimmung mit *Mbt* für Texte in neuer Rechtschreibung heraussticht. Wir verwenden hier die beiden deutschen Korpora aus *SMULTRON* als Gold-Standard und ermitteln jeweils die Anzahl der Wortformen, für die eine abweichende Wortart ermittelt wird, siehe Tabelle 6.5 auf der nächsten Seite. Die Fehlerrate ist etwa gleich

52. Siehe `http://www.ims.uni-stuttgart.de/projekte/TIGER/` (zuletzt besucht am 8.12.2010, 19:34).
53. Wir verwenden die Version 1, an einer Erweiterung der Treebank wurde zum Zeitpunkt unserer Experimente gearbeitet.
54. Es enthält 529 Sätze mit 7'416 Wortformen aus dem Roman «Sophies Welt» (Original norwegisch: Jostein Gaarder (1991) *Sofies verden: Roman om filosofiens historie*. Aschehoug) und 518 Sätze mit 10'987 Wortformen aus Presseerklärungen und Wirtschaftsberichten.

Tabelle 6.5: Fehlerrate beim Bestimmen der Wortart der deutschen Korpora aus SMULTRON *bei der Verwendung von* Mbt*, trainiert mit den aktuellsten Versionen von* NEGRA*,* TiGer *und* TüBa-D/Z*.*

Korpus	Fehler für «Sofies Welt» (7'416 Wortformen)		Wirtschaftstexte (10'987 Wortformen)	
NEGRA	6,67 %	(495)	10,29 %	(1'131)
TiGer	6,82 %	(506)	9,74 %	(1'070)
TüBa-D/Z	6,82 %	(506)	7,40 %	(813)

gross, die Fehlerquellen sind jeweils leicht verschieden. Wir sehen hier bestätigt, dass die Fehlerrate über der üblicherweise berichteten liegt – Trainingskorpus und tatsächlich behandelter Text unterscheiden sich.

Exkurs: Attribuierende Indefinitpronomen
Allerdings haben wir nach genauerer Beschäftigung mit der Annotation der Korpora *TiGer* ausgeschlossen. In der Annotierung von attribuierenden Indefinitpronomen («*kein*», «*irgendein*», «*beiden*») wird nicht zwischen attribuierenden Indefinitpronomen ohne Artikel (STTS-Tag `PIAT`) und mit Artikel (STTS-Tag `PIDAT`) unterschieden. Beide werden als `PIAT` ausgezeichnet. Diese Entscheidung können wir nicht nachvollziehen. Eine Unterscheidung beider Formen ist für Prozesse, die die Wortartenbestimmung als Grundlage verwenden, essenziell: Attribuierenden Indefinitpronomen ohne Artikel (`PIAT`) darf kein Artikel vorausgehen, etwa in «*beide Polizisten*». Attribuierenden Indefinitpronomen mit Artikel (`PIDAT`) kann jedoch ein Artikel vorausgehen, etwa in «*die beiden Polizisten*». In Beispiel 6.1 würde entsprechend *TiGer* die Wortform «*beiden*» jeweils als `PIAT` ausgezeichnet. Jedoch ist die gesamte Wortgruppe im Beispiel 6.1a Nominativ Plural, in 6.1b Akkusativ Plural, im Beispiel 6.1c jedoch Dativ Plural.

(6.1) a. Die **beiden** Polizisten halfen mir.
b. Ich sah die **beiden** Polizisten.
c. Ich half **beiden** Polizisten.

Ende des Exkurses

Wir verwenden *TüBa-D/Z* in der Version 5 als Trainingskorpus für *Mbt* – diese Entscheidung ist hauptsächlich durch den Fakt motiviert, dass *TüBa-D/Z* weiterhin aktiv gepflegt wird.

6.3.3 Zusammenfassung: Entscheidung für eine Ressource zur automatischen Wortartenbestimmung

Für die automatische Bestimmung der Wortart von Wortformen verwenden wir *Mbt*, das mit *TüBa-D/Z* trainiert wird. Der Entscheid für *Mbt* ist durch die intendierte Verwendung als interaktive Ressource begründet. Die Auswahl von *TüBa-D/Z* als Trainingskorpus ist einerseits beeinflusst durch die unbefriedigende Annotationsentscheidung für *TiGer* und andererseits motiviert durch die Tatsache, dass *TüBa-D/Z* weiterhin aktiv gepflegt und erweitert wird. Während *Mbt* eine Ressource auf dem aktuellsten Forschungsstand ist, trifft das

für die verfügbaren annotierten Korpora für Deutsch nicht zu. Daher müssen wir bezüglich der Qualität der automatischen Wortartenbestimmung gewisse Abstriche machen.

6.4 Bestimmung der Bestandteile und morphosyntaktischen Eigenschaften von Wortgruppen

Sätze bestehen aus einzelnen Wortformen, mehrere Wortformen können zu Wortgruppen[55] zusammengefasst werden. Je nach dem bestimmenden Element – Adjektiv, Substantiv, Verb etc. – oder einer syntaktischen Eigenschaft einer solchen Wortgruppe können diese weiter klassifiziert werden. Wortgruppen sind eine Struktur innerhalb natürlichsprachlicher Texte, die grösser ist als eine einzelne Wortform, jedoch kleiner als Sätze oder Teilsätze. Für die Bestimmung, welche Wortformen zu einer Wortgruppe gehören, bzw. zur Aufteilung eines Satzes in Wortgruppen werden Wortgruppentrenner (engl. *chunker*) verwendet [Abney 1991, Tjong Kim Sang und Buchholz 2000]. Wortgruppentrenner bzw. Wortgruppenextrahierer benötigen keine tiefe syntaktische Analyse der Texte, sondern verwenden die Wortarten der Wortformen – sie basieren also auf Systemen zur Wortartenbestimmung.

Für Deutsch sind in den letzten Jahren verschiedene Wortgruppentrenner entwickelt worden, etwa von Kermes und Evert [2002], Schiehlen [2002], Schmid [1995] oder Schmid und Schulte im Walde [2000]. Hinrichs [2005] bietet einen guten Überblick. Die meisten dieser Systeme dienen der Extraktion von Substantivgruppen und verwenden die von Abney [1991], Church [1988] und Ramshaw und Marcus [1995] vorgestellten Algorithmen. Komponenten wie YAC [Kermes und Evert 2002] sind für spezifische Bedürfnisse entwickelt worden und meist Teil eines umfassenderen Projektes, d. h., die Ausgabe der Ergebnisse ist für die nachfolgenden Prozesse optimiert. Da wir die Erkennung von Wortgruppen als Voraussetzung für spezifische Funktionen benötigen, sind solche Systeme für uns ungeeignet; durch die Einbindung in einen Editor müssten die Art der Eingabe und die Form der Ausgabe modifiziert werden. Zudem sind alle uns bekannten Systeme nicht für den interaktiven Einsatz gedacht. Systeme zur Extraktion von Wortgruppen sind wie morphologische Ressourcen nicht frei verfügbar (vgl. Abschnitt 6.2.3).

Wortgruppenextraktion wird hauptsächlich in Anwendungen wie Informationsextraktion und Informationsretrieval eingesetzt. Wichtig ist also die *Extraktion* entsprechender Wortgruppen. Es geht nicht darum, deren morphosyntaktische Eigenschaften zu bestimmen – abgesehen von der Kategorisierung als Substantiv- oder Verbgruppe. Solche Systeme verwenden zwar morphosyntaktische Eigenschaften der Wortformen, um zu bestimmen, welche Wortformen zu einer Wortgruppe gehören [vgl. Church 1988, Ramshaw und Marcus 1995, Schiehlen 2002], jedoch werden etwa für Substantivgruppen die Angaben zu Kasus, Genus und Numerus nicht ausgegeben.

Wir konzentrieren uns hier auf die Extraktion von Substantivgruppen und die Bestimmung ihrer morphosyntaktischen Eigenschaften. Entsprechende Informationen werden sowohl für Informations- und Bewegungsfunktionen (etwa das Hervorheben von Substantivgruppen oder das Springen zum Beginn der

55. Wir verwenden die Bezeichnung *Wortgruppe* und nicht die oft anzutreffende Bezeichnung *Phrase*, die eher aus dem englischen Sprachraum stammt und zudem den Anschein erweckt, Phrasenstrukturgrammatik würde als theoretischer Unterbau verwendet.

nächsten Substantivgruppe) als auch für Modifikationsfunktionen (das Ersetzen von Substantivgruppen durch ein Pronomen oder die Modifizierung des Kasus einer Substantivgruppe) benötigt. Ähnliche Überlegungen können natürlich für andere Wortgruppen ebenfalls angestellt werden. Wir gehen zunächst auf unsere Anforderungen an ein solches System ein (Abschnitt 6.4.1), legen anschliessend dar, auf welche Substantivgruppen wir abzielen (Abschnitt 6.4.2) und warum existierende Systeme für unsere Zwecke nicht geeignet sind. Anschliessend stellen wir in Abschnitt 6.4.3 unsere Lösung vor und berichten in Abschnitt 6.4.4 über einige Experimente zur Evaluation unseres Systems. Zudem berichten wir über Erkenntnisse bezüglich Eigenschaften von Substantivgruppen, die die Implementation von Editierfunktionen wesentlich beeinflussen.

6.4.1 Anforderungen

Neben den in Abschnitt 6.1 genannten Anforderungen hinsichtlich Verfügbarkeit, Installation, Schnittstellen und Format der Resultate, Abdeckung und Geschwindigkeit sowie Qualität der Resultate, wie sie für alle von uns verwendeten computerlinguistischen Ressourcen gelten, ergeben sich weitere Anforderungen: Der Substantivgruppenerkenner soll (a) *alle* Substantivgruppen in einem Text erkennen und (b) ihre jeweiligen morphosyntaktischen Eigenschaften hinsichtlich Genus, Kasus und Numerus (also die Kategorie einer Substantivgruppe) *korrekt* zurückliefern.

Während die zweite Forderung relativ einfach zu spezifizieren ist (die Wortformen einer Substantivgruppe sind hinsichtlich Genus, Kasus und Numerus mit dem Substantiv kongruent), ergibt sich unter Umständen eine Schwierigkeit hinsichtlich der Ambiguität von Substantivgruppen. Wir sind interessiert an möglichst eindeutigen Kategorieinformationen, um möglichst wenig Interaktion mit dem Autor zu benötigen. Die Korrektheit der gelieferten Kategorieinformationen hängt von den eingesetzten Ressourcen ab. Die Forderung nach der Korrektheit der *Erkennung* bzw. Extraktion von Substantivgruppen hängt hingegen wesentlich davon ab, wie wir *Substantivgruppe* definieren.

6.4.2 Pragmatische Definition von *Substantivgruppen*

Für viele linguistische Elemente oder Beziehungen lassen sich einerseits keine eindeutigen, allgemein akzeptierten Bezeichnungen feststellen. Andererseits ist die Bedeutung einer Bezeichnung abhängig von der zugrundeliegenden Theorie. Dies gilt auch für *Substantivgruppen*, für die wir eine pragmatische Definition geben, die wir durch die Verwendung in Funktionen zur Schreibunterstützung motivieren – wir beziehen hier den Begriff *noun phrase* mit ein, da dieser in englischsprachigen Publikationen für das von uns intendierte Phänomen verwendet wird.

Als *Substantivgruppe* betrachten wir Wortfolgen, die aus nur *einem* Substantiv und *mindestens* einer weiteren vorausgehenden Wortform bestehen; dies kann ein vorhergehendes Adjektiv (oder eine Aufzählung von Adjektiven) sein oder ein optionaler Artikel (sind Adjektive involviert, geht der Artikel den Adjektiven voraus) – ein einzelnes Substantiv ohne vorausgehende kongruente Wortformen behandeln wir als Substantiv und verwenden *GERTWOL* zur Ermittlung der Kategorie. Evert [2004] gibt einen Überblick über morphosyntaktische Eigenschaften deutscher Substantive. Typischerweise wird unsere

Variante der Substantivgruppe als *Basis-Nominalphrase* (*base noun phrase*), *nicht-rekursive Nominalphrase* (*non-recursive noun phrase*), *noun kernel* oder *contiguous noun phrase* bezeichnet. Diese Definition enspricht ebenfalls der des CoNLL-2000 Wettbewerbs [vgl. Tjong Kim Sang und Buchholz 2000].

Betrachten wir einen Beispieltext[56] und markieren die uns interessierenden Substantivgruppen:

> Der chinesische Ministerpräsident Wen Jiabao hat vor einem Rückfall in die weltweite Wirtschaftskrise gewarnt. China setzt weiter auf massive Staatsausgaben und eine lockere Geldpolitik , um seine Wirtschaft in Schwung zu halten und soziale Spannungen zu vermeiden. Wegen der Inflationsgefahr soll die stark gestiegene Kreditvergabe der Banken leicht gedrosselt werden. China sei nicht losgelöst vom Rest der Welt .

Dies entspricht einer Annotation wie in Beispiel 6.2[57].

(6.2) [$_{NP}$ Der chinesische Ministerpräsident] Wen Jiabao hat vor [$_{NP}$ einem Rückfall] in [$_{NP}$ die weltweite Wirtschaftskrise] gewarnt. China setzt weiter auf [$_{NP}$ massive Staatsausgaben] und [$_{NP}$ eine lockere Geldpolitik] , um [$_{NP}$ seine Wirtschaft] in Schwung zu halten und [$_{NP}$ soziale Spannungen] zu vermeiden. Wegen [$_{NP}$ der Inflationsgefahr] soll [$_{NP}$ die stark gestiegene Kreditvergabe] [$_{NP}$ der Banken] leicht gedrosselt werden. China sei nicht losgelöst [$_{NP}$ vom Rest] [$_{NP}$ der Welt] .

Im Deutschen können einige Präpositionen mit dem nachfolgenden bestimmten Artikel verschmelzen, wie in «*vom Rest*». Die Präposition «*von*» gehört nicht zur Substantivgruppe «*dem Rest*» – «*vom Rest*» würde von anderen Systemen als Präpositionalgruppe kategorisiert werden. Da die Wortformen «*von*» und «*dem*» miteinander verschmolzen sind, markieren wir «*vom Rest*» als Substantivgruppe und müssen in einem Nachbearbeitungsschritt diese Verschmelzung wieder trennen – dies ist mit *GERTWOL* problemlos möglich wie in Abschnitt 6.2.7.3 für «*im*» gezeigt. Possessivpronomen können die Position von unbestimmten Artikeln einnehmen, sie stimmen ebenfalls in Kasus, Genus und Numerus mit folgenden Adjektiven und Substantiven überein, wie in «*seine Wirtschaft*». Ebenso können attribuierende Indefinitpronomen oder attribuierende Demonstrativpronomen die Position des Artikels einnehmen, wie in «*solche Wechselspielchen*» oder «*dieser Trubel*».

Abhängig von theoretischen Überlegungen hinsichtlich syntaktischer Eigenschaften, ergeben sich für den obigen Text verschiedene Auszeichnungen für Substantivgruppen. So kann etwa «*die stark gestiegene Kreditvergabe der Banken*»

56. Text aus der Neuen Zürcher Zeitung, 14. März 2010, «Wen warnt vor neuer Krise», 08:39, NZZ Online http://www.nzz.ch/nachrichten/international/peking_warnt_vor_rueckfall_in_die_krise_1.5210482.html (zuletzt besucht am 8.12.2010, 19:34).
57. Wir verwenden die Auszeichnung `NP` als Kennzeichnung für Substantivgruppen, `NP` steht für *Nominalphrase* entsprechend der *STTS*-Liste.

statt zweier aufeinanderfolgender Substantivgruppen wie in Beispiel 6.3a auch als eine komplexe Substantivgruppe angesehen werden, die aus zwei kürzeren Substantivgruppen besteht, wobei letztere erstere modifiziert. Diese Substantivgruppe enthält also zwei Substantive wie in Beispiel 6.3b.

(6.3) a. [$_{NP}$ die stark gestiegene Kreditvergabe] [$_{NP}$ der Banken]
b. [$_{NP}$ [$_{NP}$ die stark gestiegene Kreditvergabe] [$_{NP}$ der Banken]]

Wir wählen hier die Zerlegung in zwei Wortgruppen ohne Kennzeichnung, dass beide syntaktisch zusammengehören, d. h., wir sind nur an nicht-rekursiven Substantivgruppen interessiert. Die Erkennung von Wortgruppen dient als Vorstufe für Informations- und Bewegungsfunktionen sowie Modifikationsfunktionen. Für eine Funktion zur Manipulation von Substantivgruppen stellt sich bei sehr komplexen Wortgruppen mit mehreren Substantiven – also rekursiven Substantivgruppen mit Einschüben – die Frage, ob wirklich alle Bestandteile von der Operation betroffen sind. Soll «*die stark gestiegene Kreditvergabe der Banken*» in ein Dativobjekt gewandelt werden, ist nur die erste Teilphrase zu modifiziern: «*der stark gestiegenen Kreditvergabe der Banken*».

Ähnliches gilt für «*eine für die Verhältnisse hohe Qualität*», hier wird eine Substantivgruppe durch einen Einschub unterbrochen:

(6.4) [$_{NP}$ eine [$_{PP}$ für [$_{NP}$ die Verhältnisse]] hohe Qualität]

Diese Einschübe können aus anderen Wortgruppen oder einzelnen Wörtern bestehen, wie in «*eine für den Anwender verständliche Sprache*» (Beispiel aus [Kermes und Evert 2002]). Anders als Kermes und Evert [2002] betrachten wir dies nicht als eine Nominalphrase mit eingeschobener Adjektivphrase entsprechend der Auszeichnung in Beispiel 6.5a, sondern als eine Substantivgruppe, in die eine Präpositionalgruppe eingeschoben wurde, entsprechend der Auszeichnung in Beispiel 6.5b.

(6.5) a. [$_{NP}$ eine [$_{AP}$ für den Anwender verständliche] Sprache]
b. [$_{NP}$ eine [$_{PP}$ für den Anwender] verständliche Sprache]

Eine Präpositionalgruppe besteht aus Präposition und Substantivgruppe, sodass in der Wortfolge «*eine für den Anwender verständliche Sprache*» zwei Basis-Substantivgruppen enthalten sind: «*eine verständliche Sprache*» und «*den Anwender*». Letztere kann sehr einfach extrahiert werden, für erstere ist zu berücksichtigen, dass zwischen Artikel und Adjektiv Wortformen auftreten, die nicht zur gesuchten Nominalphrase gehören. Komplexe Substantivgruppen müssen also weiter aufgetrennt werden, um die Basis-Substantivgruppe «*eine hohe Qualität*» oder «*eine verständliche Sprache*» zu extrahieren.

Komplexe Substantivgruppen weisen keine Beschränkungen hinsichtlich der enthaltenen Einschübe auf, wie Beispiel 6.6 zeigt.

(6.6) Nicht just die Menschheit kam in den Pfuhl und ins Loch oder eine Gruppe davon, die ihren Weg ins Himmelschreiende verderbt

> hatte, sondern nur um [$_{NP}$ **ein allerdings besonders schmuckes und überhebliches, besonders mit Prädilektion, Angelegentlichkeit und weitläufiger Planung beladenes Einzel-Vorkommnis des Geschlechtes**] handelte es sich, das man uns auf die Nase gesetzt – einem grillenhaften Gedankengang zufolge, der in den Zirkeln und Rängen nur zu wohlbekannt war und dort von jeher Bitterkeit erregt hatte, – zusammen mit der nicht ungerechtfertigten Erwartung, daß sehr bald die Bitterkeit das Teil dessen sein werde, der die kränkende Überlegung anstellte und sie ins Werk setzte.[58]

Soll eine solche Wortgruppe (Beispiel 6.7a) manipuliert werden – etwa in Definitheit und Plural (Beispiel 6.7b) oder hinsichtlich Kasus (Beispiel 6.7c) –, stellt sich heraus, dass nur einige der enthaltenen Wortformen manipuliert werden müssen.

(6.7) a. ein allerdings besonders schmuckes und überhebliches, besonders mit Prädilektion, Angelegentlichkeit und weitläufiger Planung beladenes Einzel-Vorkommnis des Geschlechtes

b. **die** allerdings besonders **schmucken und überheblichen**, besonders mit Prädilektion, Angelegentlichkeit und weitläufiger Planung **beladenen Einzel-Vorkommnisse** des Geschlechtes

c. **eines** allerdings besonders **schmucken und überheblichen**, besonders mit Prädilektion, Angelegentlichkeit und weitläufiger Planung **beladenen Einzel-Vorkommnisses** des Geschlechtes

Die eigentliche Substantivgruppe (oder Kernsubstantivgruppe) ist also «*ein schmuckes und überhebliches beladenes Einzel-Vorkommnis*». Ein System zur Wortartentrennung kann diese Nominalphrase nicht ermitteln, hierzu ist tiefe syntaktische Analyse notwendig.

Kübler et al. [2010] führen die Bezeichnung *stranded noun chunk* (`sNX`) ein, um den Artikel «*eine*» in den Beispielen 6.4 auf der vorherigen Seite und 6.5 auf der vorherigen Seite oder den Artikel «*ein*» im Beispiel 6.7a zu annotieren und so die gewünschte Substantivgruppe zu markieren. Eine solche automatische Annotation benötigt jedoch tiefe syntaktische Analyse. Und hierin liegt ein weiterer Motivationsaspekt für unsere Definition von Substantivgruppen: Die von uns intendierten Editierfunktionen werden nicht auf einen *fertigen* wohlgeformten Text in einem Nachbearbeitungs- oder Korrekturschritt angewandt, sondern auf einen *sich entwickelnden* Text. Dieser Text ist noch nicht «fertig», einige Teile sind jeweils nicht notwendigerweise wohlgeformt, vollständig oder konsistent – Van De Vanter spricht von den *three I's*: *illformedness*, *incompleteness*, *inconsistency* [Van De Vanter 1995, S. 255f]. Diese Eigenschaften machen eine korrekte tiefe syntaktische Analyse unmöglich, wie in Abschnitt 4.4 gezeigt.

Als Substantivgruppe betrachten wir also eine zusammenhängende Wortfolge bestehend aus einem Artikel, einem Adjektiv (oder einer Folge von Adjektiven) und einem Substantiv. Eine Substantivgruppe besteht aus mindestens zwei Wortformen; ist ein Artikel vorhanden, ist das Adjektiv optional, ist mindestens ein Adjektiv vorhanden, ist der Artikel optional.

58. Thomas Mann, «Joseph und seine Brüder. Der vierte Roman: Joseph, der Ernährer», Fischer Taschenbuch Verlag, Frankfurt am Main, 12. Auflage, Mai 1991, S. 11. Der Übersichtlichkeit halber ist die Substantivgruppe zusätzlich hervorgehoben.

Folgen von Adjektiven können durch Konjunktionen (gekennzeichnet durch die *STTS*-Auszeichnung `KON`) oder Adverben (`ADV`) (einschliesslich adverbial gebrauchter Adjektive) unterbrochen werden. Diese Wortformen würden bei einer Manipulation der Substantivgruppe nicht verändert, jedoch bei der Verschiebung der Nominalphrase innerhalb des Textes ebenfalls verschoben werden. Darum beziehen wir sie in die Wortfolge mit ein.

Die Rolle des Artikels kann sowohl durch bestimmte (`ART`) und unbestimmte Artikel (ebenfalls `ART`) als auch durch mit einer Präposition verschmolzene bestimmte Artikel (`APPRART`), Possessivpronomen (`PPOSAT`), attribuierende Indefenitpronomen mit und ohne Artikel (`PIDAT` und `PIAT`) und attribuierende Demonstrativpronomen (`PDAT`) gefüllt werden. Für diese Wortarten liefert die Analyse durch *GERTWOL* in der Regel Lesarten als Artikel (`DET`). Eigennamen werden nicht als Substantiv behandelt, da deren Deklination von den üblichen Regeln abweicht. Die folgende Liste zeigt einige Beispiele[59]:

(6.8) [$_{\text{ART}}$ Eine] gemischte Crew
[$_{\text{ART}}$ der] transatlantischen Fusion
[$_{\text{APPRART}}$ beim] Sozialminister
[$_{\text{PPOSAT}}$ unserem] zeitgeschichtlichen Bewusstsein
[$_{\text{PIDAT}}$ beide] Polizisten
[$_{\text{ART}}$ die] [$_{\text{PIDAT}}$ beiden] Polizisten
[$_{\text{PIAT}}$ einige] Automobilhersteller
[$_{\text{PDAT}}$ diese] heiklen Verfahren
eine [$_{\text{ADV}}$ ganz] neue Dimension
die wirtschaftliche [$_{\text{KON}}$ und] soziale Lage
das schwerste [$_{\text{ADJD}}$ natürlich] vorkommende Element

6.4.3 *NPcat* – Bestimmung von Substantivgruppen und deren Kategorie

Die exakte Definition von *Substantivgruppen* variiert in Abhängigkeit der Anwendungen, für die oder in denen Substantivgruppen verwendet werden. Beispielsweise wird im *TreeTagger chunker* eine ähnliche Definition wie unsere verwendet [vgl. Schmid und Schulte im Walde 2000]. Allerdings markiert dieser Wortgruppenextrahierer sowohl Substantiv- (`NC`) als auch Präpositionalgruppen (`PC`). Eine Präpositionalgruppe besteht hier aus einer Präposition und einer Substantivgruppe, diese wird jedoch nicht ausgezeichnet. Bei der Verwendung des *TreeTagger chunker* müssten also Präpositionalgruppen noch nachbearbeitet werden, um auch die dort enthaltenen Substantivgruppen zu extrahieren – abgesehen von der Tatsache, dass auch der *TreeTagger chunker* nur die Wortgruppen und nicht ihre Kategorien zurückliefert. *YAC* [Kermes und Evert 2002] ist dagegen für den Einsatz in der Vorverarbeitung von Korpora und Korpusanalyse gedacht und extrahiert rekursive Substantivgruppen. Beide Systeme sind nicht für den interaktiven Einsatz entwickelt worden.

Die in Wortgruppentrennern verwendeten Algorithmen sind relativ simpel [vgl. Abney 1991, Church 1988, Ramshaw und Marcus 1995] und die benötigten Ressourcen stehen uns bereits zur Verfügung: *Mbt* zur Bestimmung der Wortarten und *GERTWOL* zur morphologischen Analyse. Morphologische Analyse

59. Wenn nicht anders angegeben, stammen die folgenden Beispiele aus dem Korpus der Zeitung «Der Tagesspiegel. Zeitung für Berlin und Deutschland» von 2005 und 2006 (2'183'441 laufende Wortformen in 133'056 Sätzen).

wird in den bekannten Wortartentrennern eingesetzt, um zu bestimmen, welche Wortformen zu einer Wortgruppe gehören. Daher implementieren wir *NPcat*, welches sowohl die von uns definierten Substantivgruppen extrahiert als auch deren Kategorie zurückliefert. Die Bestimmung anderer Wortgruppen würde analog zum hier vorgestellten Ansatz funktionieren.

NPcat arbeitet in drei Schritten, die nacheinander ausgeführt werden:

1. Bestimmung der Wortart aller Wortformen eines Textes[60] mit *Mbt* (siehe Abschnitt 6.3 für Einzelheiten).

2. Extraktion der Substantivgruppen entsprechend unserer Definition in Abschnitt 6.4.2.

3. Kategorisierung aller Elemente einer Substantivgruppe mit *GERTWOL* (siehe Abschnitt 6.2) und Bestimmung der Kategorie der gesamten Substantivgruppe.

Wir stellen hier eine generelle Implementierung vor, die als alleinstehende Komponente verwendet werden kann. Darum verwenden wir *Perl* für die Schritte zwei und drei. Diese Implementierung wird auch für die Experimente in Abschnitt 6.4.4 verwendet und ist nicht explizit für den interaktiven Einsatz intendiert. Generell gilt jedoch, dass Perlprogramme schnell zu starten und auszuführen sind, die Einbindung von *NPcat* in dieser Form in andere Programme ist daher prinzipiell möglich. Zur Einbindung der Komponente in *XEmacs* (siehe Abschnitt 7.2.3) wird die Implementierung angepasst.

Als Beispiel verwenden wir den Satz 6.9:

(6.9) Nur wenn dieses strikte Verbot gelockert werde, heißt es in einer Studie der DG-Bank, könne über eine bessere Aufklärung der Verbraucher das brachliegende Potenzial konsequent erschlossen werden.

Im ersten Schritt liefert *Mbt* die Wortarten in Beispiel 6.10. Die Wortformen «*gelockert*», «*DG-Bank*» und «*Potenzial*» sind nicht im Lexikon, hier wird die Komponente zur Hypothese über unbekannte Wortformen verwendet, um Wortarten zuzuweisen. Wie in Abschnitt 6.3 bereits beschrieben, verwenden wir die Kennzeichnung entsprechend dem Stuttgart-Tübingen Tagset (STTS).

(6.10) [ADV Nur] [KOUS wenn] [PDAT dieses] [ADJA strikte] [NN Verbot] [VVPP gelockert] [VAFIN werde] [$, ,] [VVFIN heißt] [PPER es] [APPR in] [ART einer] [NN Studie] [ART der] [NN DG-Bank] [$, ,] [VMFIN könne] [APPR über] [ART eine] [ADJA bessere] [NN Aufklärung] [ART der] [NN Verbraucher] [ART das] [ADJA brachliegende] [NN Potenzial] [ADJD konsequent] [VVPP erschlossen] [VAINF werden] [$. .]

Daraus werden im zweiten Schritt mittels eines Perlprogramms die Substantivgruppen in Beispiel 6.11 auf der nächsten Seite extrahiert:

60. Die Länge des Textes ist beliebig.

(6.11) a. dieses strikte Verbot
b. einer Studie
c. der DG-Bank
d. eine bessere Aufklärung
e. der Verbraucher
f. das brachliegende Potenzial

Wortformen, die als Konjunktion oder Adverb bzw. adverbial gebrauchtes Adjektiv von *Mbt* erkannt wurden, werden zwar als Elemente der Substantivgruppe behandelt – etwa um diese anschliessend im Text zu markieren –, jedoch nicht mit *GERTWOL* analysiert, da ihre morphosyntaktischen Eigenschaften keinen Einfluss auf die Kategorie der Substantivgruppe haben. Wie in Abschnitt 6.4.2 beschrieben, werden für die Wortarten, die die Rolle des Artikels einnehmen können, jeweils die Lesarten als Artikel (`DET`) verwendet.

*Listing 6.21: Analysen für die Wortformen in «*dieses strikte Verbot*» (*GERTWOL*).*

```
dieses
(
("dieser" . [PRON MASC SG GEN])
("dieser" . [PRON NEU SG NOM])
("dieser" . [PRON NEU SG ACC])
("dieser" . [PRON NEU SG GEN])
("dieser" . [DET MASC SG GEN])
("dieser" . [DET NEU SG NOM])
("dieser" . [DET NEU SG ACC])
("dieser" . [DET NEU SG GEN])
)
strikte
(
("strikt" . [ADJ FEM SG NOM POS])
("strikt" . [ADJ FEM SG ACC POS])
("strikt" . [ADJ PL NOM POS])
("strikt" . [ADJ PL ACC POS])
("strikt" . [ADJ MASC SG NOM POS])
("strikt" . [ADJ NEU SG NOM POS])
("strikt" . [ADJ NEU SG ACC POS])
("strikt" . [ADJ FEM SG NOM POS])
("strikt" . [ADJ FEM SG ACC POS])
)
Verbot
(
("Ver|bot" . [N NEU SG NOM])
("Ver|bot" . [N NEU SG ACC])
("Ver|bot" . [N NEU SG DAT])
("ver|biet~en" . [V PAST IND SG1])
("ver|biet~en" . [V PAST IND SG3])
)
```

Für den dritten Schritt werden zunächst alle Wortformen jeder Substantivgruppe mit *GERTWOL* analysiert. Für Beispiel 6.11a liefert *GERTWOL* die Analysen in Listing 6.21[61]. Analysen für Wortarten, die *nicht* zu Substantivgruppen gehören können, werden ignoriert – hier die Lesart Pronomen (`PRON`) für «*dieser*» und die Lesart Verb (`V`) für «*Verbot*». Für diesen Schritt werden die von *Mbt* gelieferten Auszeichnungen mit den von *GERTWOL* verwendeten Bezeichnungen abgeglichen, siehe Tabelle 6.6 auf der nächsten Seite.

Anschliessend werden diese Angaben verwendet, um mittels eines Perlprogramms die Kategorie der gesamten Substantivgruppe zu ermitteln. Die Ele-

61. Wir verwenden hier die Analyse entsprechend *LSINDEX*. Aus Gründen der Lesbarkeit geben wir nur die Werte für einzelne Attribute an. Die von *GERTWOL* verwendeten Bezeichnungen für diese Werte sind in der Regel eindeutig als Werte bestimmter Attribute lesbar. Attribute, die keinen Wert haben, spielen in einer Analyse keine Rolle, siehe Abschnitt 6.2.4.3.

Tabelle 6.6: Wortartenbezeichnungen von Mbt *und* GERTWOL *(LSINDEX).*

Wortart	***Mbt***	***GERTWOL***
Artikel	ART	DET
Possessivpronomen	PPOSAT	DET und PRON
Attribuierendes Indefinitpronomen ohne Artikel	PIAT	DET und PRON
Attribuierendes Indefinitpronomen mit Artikel	PIDAT	DET und PRON
Attribuierendes Demonstrativpronomen	PDAT	DET und PRON
Präposition + Artikel	APPRART	DET
Adjektiv	ADJA	ADJ
Substantiv	NN	N

mente der Substantivgruppe müssen hinsichtlich Kasus, Genus und Numerus übereinstimmen. Das Genus von «*Verbot*» ist `NEU` in allen Lesarten als Substantiv. Darum werden die Lesarten `FEM` und `MASC` für das Adjektiv und die Lesart `MASC` für den Artikel ausgeschlossen. Die Lesarten für Artikel und Substantiv sind ausschliesslich Singular, darum können wir die Plurallesarten für das Adjektiv ausschliessen. Die Werte für Genus und Numerus sind also: Neutrum und Singular. Unter diesen Voraussetzungen bleiben nur zwei Lesarten für das Adjektiv: Nominativ und Akkusativ Singular Neutrum. Beide Lesarten sind für das Substantiv ebenfalls möglich. Die Kategorie der Substantivgruppe «*dieses strikte Verbot*» ist daher: Nominativ Neutrum Singular und Akkusativ Neutrum Singular.

Aus diesem Beispiel können wir zwei Schlussfolgerungen ziehen: (a) Da Elemente einer Substantivgruppe hinsichtlich Kasus, Genus und Numerus übereinstimmen, können wir die Schnittmenge der Kategorien dieser Elemente verwenden, um die Kategorie der Substantivgruppe zu bestimmen, siehe Formel 6.2. (b) Substantivgruppen sind hinsichtlich ihrer morphosyntaktischen Eigenschaften ambig. Wir gehen darauf in Abschnitt 6.4.4.2 genauer ein und zeigen die Konsequenzen für die Implementierung von Editierfunktionen auf.

$$Kat(DET) \cap Kat(ADJ_1) \ldots \cap Kat(ADJ_n) \cap Kat(N) \tag{6.2}$$

Einfache Substantive werden zur Bestimmung der Kategorie mit *GERTWOL* analysiert. In den meisten Fällen wird so keine eindeutige Kategorie ermittelt werden können; lediglich 7 % aller deutschen Subtantive können allein durch morphologische Analyse eindeutig einer Kategorie zugeordnet werden [Evert 2004].

6.4.4 Evaluation des Vorgehens

Die Qualität unserer Implementierung hängt von verschiedenen Faktoren ab: Ob tatsächlich alle Substantivgruppen gefunden werden, wird hauptsächlich durch die Qualität der automatischen Wortartenbestimmung beeinflusst, dazu haben wir in Abschnitt 6.3 bereits Stellung genommen. Einen weiteren Einfluss hat die Umsetzung unserer Definition einer Substantivgruppe aus Abschnitt 6.4.2. Die Qualität der Ermittlung der Kategorie der Substantivgruppen wird durch die Qualität der eingesetzten morphologischen Ressource beeinflusst, dies haben wir in Abschnitt 6.2.6 bereits erörtert.

Um festzustellen, ob unser Ansatz und die Implementierung von *NPcat* angemessen sind, haben wir verschiedene Experimente durchgeführt. Einige dieser Experimente dienen zudem dazu, einen Überblick über die Verteilung morphosyntaktischer Eigenschaften deutscher Substantivgruppen zu erhalten. Auf Grund dieser Erkenntnisse entscheiden wir, welche der prinzipiell vorstellbaren Editierfunktionen, die auf Substantivgruppen beruhen, überhaupt wert sind, implementiert zu werden.

6.4.4.1 Korrektheit der ermittelten Substantivgruppen

Zunächst stellt sich die Frage, wieviele der gewünschten Substantivgruppen entsprechend unserer Definition in Abschnitt 6.4.2 tatsächlich von *NPcat* gefunden werden und wieviele der gefundenen Substantivgruppen korrekt sind.

Wie oben schon erwähnt, hängt die Korrektheit der ermittelten Substantivgruppen hauptsächlich von der Qualität der automatischen Wortartenbestimmung mit *Mbt* ab – wenn ein Substantiv falsch als Verb klassifiziert wird, kann die entsprechende Substantivgruppe nicht mehr gefunden werden. Wir müssen hier von einer Fehlerrate von etwa 7 % ausgehen, wie in Tabelle 6.5 auf Seite 166 gezeigt. Es existieren keine annotierten Korpora, die Substantivgruppen wie von uns definiert auszeichnen, sodass wir nicht gegen einen Gold-Standard evaluieren können.

Der Einsatz von *GERTWOL* zur Ermittlung der Kategorien der Wortformen einer Substantivgruppe und die Ermittlung der Kategorie der gesamten Substantivgruppe als Schnittmenge der einzelnen Kategorien führen dazu, dass Fehler im analysierten Korpus und Fehler in der Analyse durch *Mbt* ermittelt werden können: (a) Die ermittelten Elemente einer Substantivgruppe sind nicht kongruent (siehe Beispiel 6.12a). (b) Einige Wortarten, die von *Mbt* ermittelt wurden, sind nicht korrekt, wie in Beispiel 6.12b («*kniend*» ist falsch als Substantiv erkannt worden). Da *GERTWOL* ursprünglich für den Einsatz in einer Komponente zur Rechtschreibkorrektur entwickelt wurde, werden falsch geschriebene Wortformen nicht erkannt, was zu einer Eliminierung der betroffenen Substantivgruppe führt, wie in «*das Rathuas*» («*Rathuas*» wird von *Mbt* als «unbekanntes» Wort korrekt als Substantiv markiert).

(6.12) a. * [$_{\text{NP}}$ [$_{\text{PDAT}}$ alle] [$_{\text{NN}}$ Auto]]
b. * [$_{\text{NP}}$ [$_{\text{PIDAT}}$ allen] [$_{\text{ADJA}}$ vieren] [$_{\text{NN}}$ kniend]]

6.4.4.2 Morphosyntaktische Eigenschaften deutscher Substantivgruppen

Die Bestimmung der Kategorien von komplexen Substantivgruppen wird für darauf aufbauende Funktionen benötigt. Wenn die Kategorie einer Substantivgruppe ambig ist, kann dies abhängig von der eigentlichen Funktion unter Umständen vernachlässigt werden. Falls tatsächlich die exakte Kategorie benötigt wird, kann diese nur über Interaktion mit dem Benutzer bestimmt werden. Sollte sich herausstellen, dass für Substantivgruppen lediglich etwa 20 % eindeutig kategorisiert werden können, wie von Evert [2004] gezeigt, müsste man den Schluss ziehen, dass sich die Verwendung von Substantivgruppen als Element in Editierfunktionen – und somit die Implementierung von auf diesen

operierenden Funktionen – nicht lohnt. Unsere Editierfunktionen sollen ja die kognitive Belastung von Autoren reduzieren. Dieser Effekt wird unter Umständen ins Gegenteil gekehrt, wenn der Autor jeweils erst noch mit der Funktion interagieren und die korrekte Kategorie auswählen muss.

Wir haben *NPcat* auf unser Tagesspiegel-Korpus angewandt. Dieses besteht aus Artikeln von 2005 und 2006 mit insgesamt 2'183'441 laufenden Wortformen (133'056 Sätze). *NPcat* kann daraus 515'424 Substantivgruppen ermitteln, wobei 153'200 lediglich aus einem einzelnen Substantiv bestehen und darum nach dem ersten Schritt ausgeschlossen wurden. Von den 244'958 distinkten Substantivgruppen bestehen 44'688 aus nur einem Substantiv. Tabelle 6.7 zeigt die Anzahl der ermittelten Kategorien der Substantivgruppen.

Total	Unbekannt	1 Kategorie	2 Kategorien	3 Kategorien	4 und mehr
Alle SG 362'224	18'359 (5,07 %)	136'595 (37,71 %)	181'887 (50,21 %)	7'742 (2,14 %)	17'641 (4,87 %)
Dist. SG200'270	15'888 (7,93 %)	71'545 (35,73 %)	94'994 (47,43 %)	4'646 (2,32 %)	13'197 (6,59 %)

Tabelle 6.7: Kategorien von Substantivgruppen (SG).

Von den 362'588 Substantivgruppen konnte für 18'359 (5,07 % der Substantivgruppen) keine Kategorie ermittelt werden. Als «unbekannt» werden von *NPcat* folgende Fälle markiert:

- Es ist keine Kongruenz zwischen den Elementen einer Substantivgruppe festzustellen. Dies betrifft 2'663 Substantivgruppen (0,74 %). In einigen Fällen, wie in «*alle Auto*», ist es gar keine Substantivgruppe («*Und deshalb wird es für **alle Auto** fahrenden Benutzer von Mobiltelefonen …*»). Hier handelt es sich also nicht um einen Fehler von *NPcat*. In Fällen wie «*ihre tiefsten Krise*» handelt es sich um einen Fehler im Ausgangstext (der entsprechende Satz heisst «*Die ‹Affäre Kohl›, die dessen langjährigen Kronprinzen zum Übergangsvorsitzenden degradierte und der Partei **ihre tiefsten Krise** bescherte, hatte damit vor einem Jahr erst so richtig begonnen.*»).

- Die von *Mbt* ermittelte Wortart ist falsch, wie etwa in «*kniend*», das als Substantiv erkannt wurde. Hier wird also ein Fehler von *Mbt* durch *GERTWOL* korrigiert.

- Eine Wortform enthält einen Tippfehler und wird darum von *GERTWOL* nicht erkannt, obwohl die von *Mbt* ermittelte Wortart als Hypothese unter Einbeziehung des Schreibfehlers korrekt ist, wie für «*Rathuas*» oder «*Baulöwens*»[62]. Hier ist unklar, ob wir dies als Fehler betrachten wollen.

- Die ermittelten Elemente der Substantivgruppe sind korrekt, jedoch werden einige Wortformen von *GERTWOL* nicht analysiert, wie «*schwächelnden*» in «*der schwächelnden US-Konjunktur*», oder falsch analysiert, wie «*viele*» als attribuierendes Indefinitpronomen ohne Artikel. Hier handelt es sich in der Tat um einen Fehler.

136'595 Substantivgruppen (37,71 %) konnten eindeutig kategorisiert werden, 181'887 Substantivgruppen (50,21 %) sind zweifach ambig, es wurden zwei Kategorien ermittelt. Einen detaillierten Überblick über diese beiden Klassen gibt

62. Vermutlich handelt es sich um einen Genitiv, die korrekte Form wäre jedoch «*Baulöwen*», die falsche Form wird von *GERTWOL* nicht analysiert.

Tabelle 6.8: Morphosyntaktische Ambiguität von Substantivgruppen (laufende Wortformen).

Anzahl Substantivgruppen	% Substantivgruppen	Kategorie
23'940	6,60	MASC SG NOM
8'130	2,24	MASC SG GEN
31'086	8,57	MASC SG DAT
18'994	5,24	MASC SG ACC
3'195	0,88	MASC PL GEN
4'849	1,34	MASC PL DAT
3	0,00	FEM SG NOM
53	0,01	FEM SG GEN
3'727	1,03	FEM SG DAT
3'205	0,88	FEM PL GEN
4'125	1,14	FEM PL DAT
1	0,00	NEU SG NOM
6'427	1,77	NEU SG GEN
20'819	5,74	NEU SG DAT
1'843	0,51	NEU SG ACC
2'196	0,61	NEU PL GEN
3'797	1,05	NEU PL DAT
66	0,02	nil PL DAT
106	0,03	nil PL GEN
33	0,01	nil nil nil
6	0,00	MASC SG NOM, NEU SG DAT
5'113	1,41	MASC SG NOM, MASC PL GEN
23	0,01	MASC SG NOM, FEM PL GEN
109	0,03	MASC SG NOM, NEU PL GEN
1	0,00	MASC SG NOM, nil PL GEN
21	0,01	MASC SG GEN, MASC SG ACC
1'068	0,29	MASC SG GEN, NEU SG GEN
2	0,00	MASC SG GEN, NEU SG DAT
5	0,00	MASC SG GEN, MASC PL DAT
1'251	0,35	MASC SG DAT, MASC SG ACC
3'942	1,09	MASC SG DAT, NEU SG DAT
7	0,00	MASC SG ACC, MASC PL GEN
1'589	0,44	MASC SG ACC, MASC PL DAT
40	0,01	MASC SG ACC, FEM PL DAT
3	0,00	MASC SG ACC, NEU PL DAT
604	0,17	MASC SG ACC, nil PL DAT
15'258	4,21	MASC PL NOM, MASC PL ACC
2	0,00	MASC PL GEN, FEM SG GEN
123	0,03	MASC PL GEN, FEM PL GEN
68	0,02	MASC PL GEN, NEU PL GEN
44	0,01	MASC PL GEN, nil PL GEN
669	0,18	MASC PL DAT, NEU SG DAT
138	0,04	MASC PL DAT, FEM PL DAT
209	0,06	MASC PL DAT, NEU PL DAT
90	0,02	MASC PL DAT, nil PL DAT
58'197	16,05	FEM SG NOM, FEM SG ACC
41'763	11,52	FEM SG GEN, FEM SG DAT
6	0,00	FEM SG GEN, nil PL GEN
12'798	3,53	FEM PL NOM, FEM PL ACC
52	0,01	FEM PL GEN, NEU PL GEN
4	0,00	FEM PL GEN, nil PL GEN
3	0,00	FEM PL DAT, NEU SG DAT
168	0,05	FEM PL DAT, NEU PL DAT
3	0,00	FEM PL DAT, nil PL DAT
2	0,00	NEU SG NOM, NEU SG DAT
29'495	8,13	NEU SG NOM, NEU SG ACC
13	0,00	NEU SG GEN, NEU SG DAT
10	0,00	NEU SG GEN, NEU PL DAT
252	0,07	NEU SG DAT, NEU PL DAT
1	0,00	NEU SG DAT, nil PL NOM
7'074	1,95	NEU PL ACC, NEU PL NOM
1'661	0,46	nil PL ACC, nil PL NOM

Tabelle 6.8 auf der vorherigen Seite[63]. Eventuell ist es möglich, die Anzahl der ambigen Substantivgruppen weiter zu reduzieren, wenn man die Valenzrahmen des Verbs im jeweiligen Satz berücksichtigt. Dies würde jedoch wiederum tiefe syntaktische Analyse voraussetzen. Zudem steht in Nebensätzen das finite Verb an letzter Stelle – es ist also möglich, dass es zum Zeitpunkt des Aufrufs einer Editierfunktion noch gar nicht geschrieben ist.

Betrachten wir nicht laufende Wortformen im Korpus des «Tagesspiegels», sondern nur distinkte von uns extrahierte Substantivgruppen, können wir 200'270 Substantivgruppen ermitteln. Für 15'888 Substantivgruppen (7,93 %) kann keine Kategorie ermittelt werden: Für 2'243 Substantivgruppen kann keine *gemeinsame* Kategorie ermittelt werden, also keine Kongruenz festgestellt werden. In 13'639 Fällen wird mindestens eine der Wortformen von *GERTWOL* nicht erkannt, oder die von *GERTWOL* ermittelte Wortart stimmt nicht mit der von *Mbt* ermittelten Wortart überein. Die Gründe sind identisch mit den Gründen für fehlende Analysen für Substantivgruppen im laufenden Text. 71'545 Substantivgruppen (35,72 %) können eindeutig kategorisiert werden, für 94'994 Substantivgruppen (47,43 %) können zwei Kategorien ermittelt werden. Im Fall von distinkten Substantivgruppen haben also 83,15 % maximal zwei Kategorien[64]. Einen genauen Überblick über diese beiden Klassen gibt Tabelle 6.9 auf der nächsten Seite.

Dies ist ein sehr vielversprechendes Resultat. Für ein Drittel aller Substantivgruppen ist keine Interaktion notwendig, da die exakte Kategorie eindeutig bestimmt werden kann. Die von uns ermittelten Werte sind besser als die von Evert [2004] berichteten, allerdings wurde dort das Genus nicht einbezogen. Für etwa die Hälfte aller Substantivgruppen (diejenigen mit zwei Kategorien) hängt die Notwendigkeit zur genauen Bestimmung der Kategorie von der Funktion ab, die diese Substantivgruppen verwendet. Für die Pluralisierung einer Substantivgruppe müssen beispielsweise Genus und Kasus der Originalwortgruppe beibehalten werden, lediglich der Numerus ändert sich. Die folgenden Beispiele machen die Notwendigkeit zur Interaktion, abhängig von den konkreten Kategorien, deutlich:

- Die Substantivgruppe «*die weltweite Wirtschaftskrise*» ist sowohl `FEM SG NOM` als auch `FEM SG ACC`. Die Pluralformen sind oberflächengleich: «*die weltweiten Wirtschaftskrisen*». 58'197 aller Substantivgruppen (16,05 %) werden als `FEM SG NOM` und `FEM SG ACC` kategorisiert.
- Für «*das Konzerthaus*» können wir zwei Kategorien ermitteln: `NEU SG NOM` und `NEU SG ACC`. Auch hier sind Pluralformen beider Kategorien oberflächengleich («*die Konzerthäuser*») – hier muss die Ambiguität also nicht aufgelöst werden. 29'495 aller Substantivgruppen (8,13 %) in unserem Tagesspiegelkorpus wurden kategorisiert als `NEU SG NOM` und `NEU SG ACC`.
- Für «*der Reparaturwerkstatt*» erhalten wir die beiden Kategorien `FEM SG GEN` und `FEM SG DAT`. Die Pluralformen dieser beiden Kategorien sind

63. Für Werte, die in der Tabelle mit `nil` angegeben sind, gilt, dass sie jeweils alle Formen an dieser Position annehmen können, also hinsichtlich dieses Attributs ambig sind. Für die erste Position wäre das `MASC/FEM/NEU`, für die zweite Position `SG/PL` und für die dritte Position `NOM/GEN/DAT/ACC`.
64. Natürlich gilt diese Aussage nicht für deutsche Substantivgruppen allgemein, sondern so exakt nur für unser Tagesspiegel-Korpus.

Tabelle 6.9: Morphosyntaktische Ambiguität distinkter Substantivgruppen.

Anzahl Substantivgruppen	% Substantivgruppen	Kategorie
13'279	6,63	MASC SG NOM
5'033	2,51	MASC SG GEN
13'680	6,83	MASC SG DAT
10'130	5,06	MASC SG ACC
2'106	1,05	MASC PL GEN
3'241	1,62	MASC PL DAT
3	0,00	FEM SG NOM
1'977	0,99	FEM SG DAT
51	0,03	FEM SG GEN
2'322	1,16	FEM PL GEN
2'665	1,33	FEM PL DAT
1	0,00	NEU SG NOM
3'367	1,68	NEU SG GEN
9'550	4,77	NEU SG DAT
817	0,41	NEU SG ACC
1'204	0,60	NEU PL GEN
2'017	1,01	NEU PL DAT
39	0,02	nil PL DAT
61	0,03	nil PL GEN
2	0,00	nil nil nil
5	0,00	MASC SG NOM, NEU SG DAT
2'010	1,00	MASC SG NOM, MASC PL GEN
1	0,00	MASC SG NOM, FEM PL GEN
61	0,03	MASC SG NOM, NEU PL GEN
1	0,00	MASC SG NOM, nil PL GEN
20	0,01	MASC SG GEN, MASC SG ACC
674	0,34	MASC SG GEN, NEU SG GEN
2	0,00	MASC SG GEN, NEU SG DAT
5	0,00	MASC SG GEN, MASC PL DAT
1'160	0,58	MASC SG DAT, MASC SG ACC
1'799	0,90	MASC SG DAT, NEU SG DAT
7	0,00	MASC SG ACC, MASC PL GEN
806	0,40	MASC SG ACC, MASC PL DAT
4	0,00	MASC SG ACC, FEM PL DAT
3	0,00	MASC SG ACC, NEU PL DAT
312	0,16	MASC SG ACC, nil PL DAT
8'638	4,31	MASC PL NOM, MASC PL ACC
2	0,00	MASC PL GEN, FEM SG GEN
57	0,03	MASC PL GEN, FEM PL GEN
59	0,03	MASC PL GEN, NEU PL GEN
22	0,01	MASC PL GEN, nil PL GEN
519	0,26	MASC PL DAT, NEU SG DAT
97	0,05	MASC PL DAT, FEM PL DAT
138	0,07	MASC PL DAT, NEU PL DAT
49	0,02	MASC PL DAT, nil PL DAT
29'487	14,72	FEM SG NOM, FEM SG ACC
19'975	9,97	FEM SG GEN, FEM SG DAT
5	0,00	FEM SG GEN, nil PL GEN
8'672	4,33	FEM PL NOM, FEM PL ACC
32	0,02	FEM PL GEN, NEU PL GEN
3	0,00	FEM PL GEN, nil PL GEN
3	0,00	FEM PL DAT, NEU SG DAT
95	0,05	FEM PL DAT, NEU PL DAT
3	0,00	FEM PL DAT, nil PL DAT
2	0,00	NEU SG NOM, NEU SG DAT
15'005	7,49	NEU SG NOM, NEU SG ACC
13	0,01	NEU SG GEN, NEU SG DAT
9	0,00	NEU SG GEN, NEU PL DAT
221	0,11	NEU SG DAT, NEU PL DAT
1	0,00	NEU SG DAT, nil PL NOM
4'375	2,18	NEU PL NOM, NEU PL ACC
642	0,32	nil PL NOM, nil PL ACC

«*der Reparaturwerkstätten*» und «*den Reparaturwerkstätten*» – hier muss also disambiguiert werden. 41'763 aller Substantivgruppen (8,1 %) werden kategorisiert als `FEM SG GEN` und `FEM SG DAT`.

Auf der Basis unserer Experimente und der Art der von uns intendierten Editierfunktionen schätzen wir, dass in etwa 60 % der Fälle keine Interaktion zur exakten Bestimmung der Kategorie einer Substantivgruppe notwendig ist – weil sie entweder eindeutig ermittelt werden kann oder die Ambiguität nicht aufgelöst werden muss. Unseres Wissens gibt es momentan keine Untersuchungen, die sich mit der Verteilung morphosyntaktischer Eigenschaften von Wörtern oder Wortgruppen im Deutschen beschäftigen. Kübler et al. [2010] berichten einige Zahlen zu eingebetteten Adjektivgruppen in Substantivgruppen im *TüBa-D/Z*-Korpus, doch dies ist lediglich ein Anfang.

6.4.4.3 Qualität der Kategoriebestimmung von Substantivgruppen

Schliesslich stellt sich die Frage nach der Korrektheit der ermittelten Kategorien für die Substantivgruppen. Neben dem Aspekt, ob Autoren für eine bestimmte Funktion die exakte Kategorie einer Substantivgruppe aus einer Liste auswählen müssen, ist wichtig, wie vertrauenswürdig die vom System ermittelten Kategorien sind. Ist die Qualität ungenügend, müssten Autoren alle betroffenen Textstellen, die von einer Funktion geändert worden wären, kontrollieren und möglicherweise selbst ändern – womit der intendierte Effekt der Erleichterung des Redigierens zunichte gemacht wäre.

Für Kategorien, die nur sehr selten (weniger als 10 mal) vorkommen, ist in Tabelle 6.10 auf der nächsten Seite[65] aufgelistet, welche Substantivgruppe so analysiert wurde. Der Verdacht liegt nahe, dass es sich um Fehler handelt, wenn einzelne Kategorien nur äusserst selten auftreten oder bei ambigen Substantivgruppen die Kombination der Kategorien sehr ungewöhnlich ist, insbesondere hinsichtlich Genus. Diese stichprobenartige Untersuchung der Korrektheit der ermittelten Kategorien bestätigt die oben getroffenen Aussagen zu Fehlerquellen in *Mbt* und *GERTWOL*. Einzelne Fehler oder ungewöhnliche Kategorisierungen sind auf fehlerhafte Lexikoneinträge von *GERTWOL* zurückzuführen (fehlende Einträge, falsch kategorisierte Einträge oder Einträge von heute ungebräuchlichen Wörtern).

Um die Frage nach der Qualität der Kategorisierung genauer zu beantworten, haben wir zwei Stichproben ausgewertet – eine von den eindeutig kategorisierten und eine von den zweifach ambigen Substantivgruppen der distinkten Substantivgruppen. Jede Stichprobe enthält 384 Substantivgruppen, um ein Konfidenzniveau von 95 % mit 5 % Stichprobenfehlern zu gewährleisten, entsprechend der Formel 6.1 auf Seite 148. Das Resultat ist in Abbildung 6.8 auf Seite 183 dargestellt.

Für eindeutige Kategorien ist in 99,48 % der Fälle diese Kategorie auch korrekt, es treten nur zwei Fehler auf:

(6.13) a. * deren Freundin: `FEM SG GEN`
b. * deren Schwester: `FEM SG GEN`

65. **N** ist die Anzahl Vorkommen dieser Kategorie im Tagesspiegel-Korpus.

Kategorie	N	Substantivgruppe mit Wortarten	Bemerkung
FEM SG NOM	3	die/ART meisten/PIDAT türkischer/ADJA Herkunft/NN; häufigsten/ADJA formulierte/ADJA Gegenargument/NN; ihre/PPOSAT tiefsten/ADJA Krise/NN	Falsche Analyse
NEU SG NOM	1	häufigsten/ADJA formulierte/ADJA Gegenargument/NN	
MASC SG NOM, NEU SG DAT	6	bayrischer/ADJD oder/KON baden-württembergischer/ADJA Kollege/NN; geltender/ADJD Stuttgarter/ADJA Kollege/NN; iranischer/ADJA Kollege/NN; sadistischer/ADJA Kollege/NN; schottischer/ADJA Kollege/NN; sächsischer/ADJA Kollege/NN	Die Analysen sind richtig; «*Kollege*» wird auf die Grundformen «*der Kollege*» und «*das Kolleg*» zurückgeführt
MASC SG NOM, FEM PL GEN	23	der/ART Westen/NN	korrekt («*die Weste*», «*der Westen*»)
MASC SG NOM, nil PL GEN	1	der/ART Personalkosten/NN	Falsche Zerlegung «*Person+alk+osten*»
MASC SG GEN, NEU SG DAT	2	bestehenden/ADJA Jakob-Kaiser-Haus/NN; gastfreien/ADJA Haus/NN	Analyse von «*Haus*» als Genitiv von «*Hau*»
MASC SG GEN, MASC PL DAT	5	allen/PIDAT Balkons/NN; allen/PIDAT Tricks/NN; allen/PIDAT möglichen/ADJA Tests/NN; diesen/PDAT Codes/NN; diesen/PDAT Jobs/NN	Analyse von «*allen*» als `MASC SG GEN` ist nur für Abstrakta möglich, aber dies kann erst auf der semantischen Ebene ermittelt werden
MASC PL GEN, MASC SG ACC	7	manchen/PIAT Autofahrer/NN; manchen/PIAT Bauchnabel/NN; manchen/PIAT Käufer/NN; manchen/PIAT Lacher/NN; manchen/PIAT Privatanleger/NN; manchen/PIAT Zuhörer/NN; manchen/PIAT Zulieferer/NN	falsche Analyse von «*manchen*», korrekt wäre nur `MASC SG ACC`
MASC SG ACC, FEM PL DAT	40	den/ART Kugelwesten/NN; den/ART Westen/NN; den/ART gesamten/ADJA Westen/NN; seinen/PPOSAT Husky-Westen/NN	korrekt
MASC SG ACC, NEU PL DAT	3	den/ART Dekan/NN; den/ART Keks/NN; den/ART Massakern/NN	falsche Analysen von *GERTWOL*
FEM SG GEN, FEM PL GEN	2	der/ART Carens/NN; der/ART Lisas/NN	falsch, Eigennamen als Substantiv markiert
FEM SG GEN, NIL PL GEN	6	deren/PDAT Verlobten/NN; deren/PDAT Vorsitzender/NN; deren/PDAT Vorstandsvorsitzenden/NN; deren/PDAT Vorstandsvorsitzender/NN; deren/PDAT Angehörigen/NN	falsche Analysen von *GERTWOL* für die Substantive; zusätzlich ein Fehler von *Mbt*: «*deren*» hätte jeweils als `PRELAT` und nicht als `PDAT` markiert werden müssen
FEM PL GEN, nil PL GEN	5	der/ART Lebensbedingungen/NN; der/ART Rechtsstreitigkeiten/NN; der/ART Wirtschaftsbeziehungen/NN	falsch, Substantive auch als *Pluralia Tantum* von *GERTWOL* erkannt
FEM PL DAT, NEU SG DAT	3	anderen/ADJA Tierhäuten/NN; formvollendeten/ADJA Brüsten/NN; großen/ADJA Brüsten/NN	falsch, Substantiv jeweils auch als substantivierter Infinitiv analysiert
FEM PL DAT, nil PL DAT	3	diesen/PDAT beiden/PIDAT Alternativen/NN; den/ART Lebensbedingungen/NN; den/ART Streitigkeiten/NN	falsch
NEU SG DAT, NEU SG NOM	2	größten/ADJA Museumsprojekt/NN; sklavischsten/ADJA Land/NN	falsche Analyse des Adjektivs als Elativ-Form
NEU SG GEN, NEU PL DAT	10	allen/PIDAT entscheidenden/ADJA Details/NN; allen/PIDAT Hotels/NN; allen/PIDAT Ressorts/NN; allen/PIDAT Milieus/NN; allen/PIDAT strittigen/ADJA Details/NN; allen/PIDAT Genres/NN; allen/PIDAT Cafés/NN; allen/PIDAT Details/NN; allen/PIDAT neuen/ADJA Cabrios/NN	falsche Analyse von *GERTWOL* für «*allen*» als attribuierendes Indefinitpronomen für abstrakte *Singularia Tantum* wie «*Müll*»
NEU SG DAT, nil PL NOM	1	angehauchten/ADJA Halbdunkel/NN	falsch

Tabelle 6.10: Seltene Kategorien für Substantivgruppen.

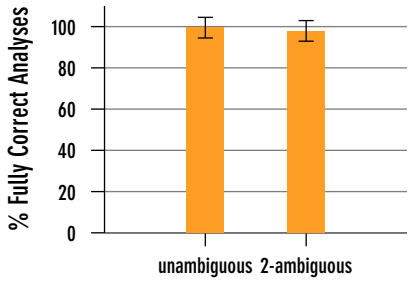


Abbildung 6.8: Prozentzahl der vollständig korrekten Analysen von NPcat *(Konfidenzintervall von 5 %).*

In beiden Fällen ist «*deren*» von *Mbt* falsch als attribuierendes Demonstrativpronomen (`PDAT`) erkannt worden. Tatsächlich ist «*deren*» ein attribuierendes Relativpronomen (`PRELAT`). Dieses Problem kann durch verbessertes Training des Wortartenerkenners behoben werden. Beide Beispiele sind hinsichtlich Kasus ambig – also `NOM/GEN/DAT/ACC`. Die Ambiguität der Substantive wird von *GERTWOL* auch korrekt erkannt.

Die falschen Kategorien für zweifach ambige Substantivgruppen – 97,92 % sind korrekt kategorisiert – werden durch ungebräuchliche (jedoch prinzipiell korrekte) Analysen der Substantive durch *GERTWOL* verursacht, siehe Beispiel 6.14. Würde *GERTWOL* Gewichtung für die Analysen verwenden, würden wenig plausible Dekompositionen wie für «*Flugzeuge*» als «*der Flug-Zeuge*» (Beispiel 6.14a) oder «*Urteil*» als «*der Ur-Teil*» (Beispiel 6.14b) vermieden. Ebenso würden Lesarten als substantivierte Verbalisierungen wie für «*Hauptsätzen*» als «*das Haupt-Sätzen*» (Beispiel 6.14e) nicht als Resultat verwendet.

(6.14)
a. * der Zivilflugzeuge: `MASC SG NOM, NEU PL GEN`
b. * seinem Urteil: `MASC SG DAT, NEU SG DAT`
c. * vielen Straßenkämpfen: `MASC PL DAT, NEU SG DAT`
d. * möglichen Punkten: `MASC PL DAT, NEU SG DAT`
e. * kurzen Hauptsätzen: `MASC PL DAT, NEU SG DAT`

6.4.5 Zusammenfassung: Erkennung und Kategorisierung von Wortgruppen

NPcat, unsere Kombination aus *Mbt* und *GERTWOL*, liefert bei der Erkennung von Substantivgruppen Ergebnisse, die vergleichbar mit dedizierten Chunkparsern wie *YAC* [Kermes und Evert 2002] sind, bietet jedoch den Vorteil, dass keine zusätzlichen Ressourcen notwendig sind, es interaktiv verwendet werden kann und die Kategorie der Subtantivgruppe zurückliefert. *NPcat* ist eine gleichwertige Implementierung üblicher Algorithmen wie von [Schiehlen 2002] beschrieben.

6.5 Zusammenfassung

In diesem Kapitel haben wir uns mit computerlinguistischen Systemen für Deutsch beschäftigt. Für die Umsetzung des in Kapitel 4 vorgestellten Konzepts von linguistisch motivierten Editierfunktionen benötigen wir geeignete Ressourcen zur Analyse und Generierung von Wortformen sowie zur flachen syntaktischen Analyse.

Seit dem ersten Wettbewerb für morphologische Systeme für Deutsch 1994 wurde keine umfassende Evaluation solcher Komponenten durchgeführt, es wurden seitdem jedoch neue Systeme entwickelt. Wir konnten uns daher nicht auf bereits vorliegende Daten zu Eigenschaften entsprechender Systeme stützen. In Abschnitt 6.2 haben wir *Stripey Zebra*, *GERTWOL/GERGEN*, *Morphisto* und *mOLIFde* hinsichtlich der in Abschnitt 6.1 genannten Anforderungen zur Verfügbarkeit, zum Installieren/Kompilieren, zur Abdeckung sowie zur Weiterverarbeitbarkeit und Qualität der Resultate evaluiert. Es hat sich herausgestellt, dass keines der Systeme unseren Anforderungen vollumfänglich entspricht. *GERTWOL/GERGEN* in der aktuellen Version ist am ehesten geeignet, auch wenn wir verschiedene Fehlerquellen ausmachen, die die Qualität im Vergleich mit früheren Versionen mindern.

Für die automatische Wortartenbestimmung haben wir uns für *Mbt* entschieden und verwenden *Tüba-D/Z* zum Trainieren. Die momentan verfügbaren annotierten Korpora für Deutsch entsprechen leider nicht den aktuellen Rechtschreibregeln, sodass wir hier bezüglich der Qualität der Resultate ebenfalls Abstriche machen müssen.

Für die flache syntaktische Analyse haben wir *NPcat* implementiert, um die Elemente von Substantivgruppen und deren morphosyntaktische Eigenschaften zu bestimmen. Eine eigene Entwicklung ist hier notwendig, da die verfügbaren Systeme jeweils nur die Elemente von Wortgruppen ermitteln, jedoch nicht die Kategorie. Aus den Daten, die wir während der Evaluation von *NPcat* gewonnen haben, konnten wir zusätzlich Angaben zur Ambiguität deutscher Substantivgruppen ermitteln. Solche Informationen liegen unseres Wissens bislang noch nicht vor. Angaben zur Ambiguität von Wortgruppen oder Wortarten sind notwendig, um einzuschätzen, wie intensiv die Interaktion mit dem Benutzer sein muss, der eine Funktion aufruft, die auf diesen Einheiten operiert.

Analog verfügbarer Informationen zur Ambiguität von Substantiven und der von uns ermittelten Eigenschaften von Substantivgruppen, wären Informationen zur Ambiguität von Verbformen nützlich. Wir finden in der Literatur ebenfalls noch keine Angaben zur Verteilung von Flexionsklassen innerhalb verschiedener Wortarten. Hier sind für die Korpuslinguistik noch viele Fragen offen.

Insgesamt müssen wir festhalten, dass unsere Annahme A.3 aus Abschnitt 1.2, computerlinguistische Ressourcen wären frei verfügbar und für die Integration in praxisrelevante Applikationen geeignet, sich nicht bestätigt hat. Angesichts aktueller Aktivitäten der «Gesellschaft für Sprachtechnologie und Computerlinguistik» zur Ausrichtung von Wettbewerben und des Erfolgs des von uns gemeinsam mit Michael Piotrowski ausgerichteten ersten «Workshop on Systems and Frameworks for Computational Morphology» 2009 [Mahlow und Piotrowski 2009b] ist jedoch zu vermuten, dass das Bedürfnis nach qualitativ hochwertigen Systemen für Deutsch wächst und entsprechende Entwicklungsarbeit geleistet wird.

In aktuellen Call-for-Papers werden explizit die Angabe und Abgabe von verwendeten Ressourcen oder implementierten Systemen gefördert bzw. gefordert: etwa für das «49th Annual Meeting of the Association for Computa-

tional Linguistics: Human Language Technologies» 2011[66] oder für die «12th International Conference on Intelligent Text Processing and Computational Linguistics» (CICLing) 2011[67]. Es wird sich hoffentlich durchsetzen, genauer anzugeben, womit gearbeitet wurde und wie bestimmte Resultate zustande kommen, wie wir es in Abschnitt 6.2.8 kritisiert haben.

66. «[…] papers accompanied by the resource (software or data) described in the paper […] will also be reviewed for the quality of the resource that is being made available. Papers that are submitted with accompanying software/data will receive additional credit toward the overall evaluation score, and acceptance or rejection decision will be made based on the quality of both the research and the software/data component.» http://www.acl2011.org/call.shtml (zuletzt besucht am 8.12.2010, 19:34)

67. «Starting from this year, CICLing start implementing a new policy of giving preference to papers with verifiable and reproducible results: If the authors claim to have obtained some results, we encourage them to make all the input data necessary to verify and reproduce the results available to the community. If the authors claim to advance human knowledge by introducing an algorithm, we encourage them to make the algorithm itself—and not only its (usually vague and incomplete) description—available to the public.» http://cicling.org/2011/right.htm#Verifiability (zuletzt besucht am 8.12.2010, 19:34)

7 Umsetzung

It is very important that the user feel in control of the editor and not the other way around.

—Steven R. Wood, Z – *The 95 % Program Editor*, 1981

This is the proper use of an imperfect rule-driven program: its output is expected to be examined by a person.

—Richard Hamlet, *A Disciplined Text Environment*, 1986

In Kapitel 4 haben wir das Konzept des linguistisch unterstützten Redigierens vorgestellt und im vorigen Kapitel 6 computerlinguistische Ressourcen für die Implementierung entsprechender Funktionen ausgewählt. In diesem Kapitel zeigen wir die prototypische Implementierung solcher Funktionen.

Zunächst stellen wir im Abschnitt 7.1 den verwendeten Editor *XEmacs* vor. Wir gehen auf bereits vorhandene Unterstützung für Autoren natürlichsprachlicher Texte sowie auf allgemeine Prinzipien in der Programmierung von *XEmacs* ein, die für unser Projekt relevant sind. Anschliessend stellen wir in Abschnitt 7.2 vor, wie die in Kapitel 6 ausgewählten Ressourcen in *XEmacs* eingebunden werden können. In Abschnitt 7.3 zeigen wir die konkrete Umsetzung ausgewählter Funktionen. Diese Funktionen stehen beispielhaft für die in Abschnitt 4.2 gezeigten Funktionstypen. In Abschnitt 7.4 zeigen wir dann, wie die Implementierung weiterer Funktionen aus diesen Beispielen abgeleitet werden kann, um eine möglichst breite Abdeckung der vorgeschlagenen Funktionen zu erreichen.

7.1 *XEmacs* als Editor

XEmacs ist eine Variante des Editors *Emacs*[1]. *Emacs* wurde Ende der 1970er Jahre am MIT von Richard M. Stallman [1981] entwickelt. Die Bezeichnung der aktuellen Version der ursprünglichen Implementierung ist *GNU Emacs*, da eine GNU GPL Lizenz verwendet wird. *Emacs* ist als Texteditor entwickelt worden und bis auf den Kern in Lisp implementiert. Von Anfang an war die Absicht, dem Benutzer zu erlauben, jederzeit Funktionen hinzuzufügen und vorhandene Funktionen zu ändern. Lediglich von der Anwendung direkt interpretierte Sprachen wie Lisp erlauben es, *während* der Laufzeit Code zu ändern und zu integrieren, ohne das gesamte Programm neu kompilieren und anschliessend neu starten zu müssen.

Die ursprünglichen Prinzipien finden wir heute in allen Varianten des *Emacs*: (a) Der Quellcode ist vollständig zugänglich. (b) Jeder Benutzer kann jede vorhandene Funktion hinsichtlich der verwendeten Parameter, des Skopus, des Funktionsnamens, der Ausführung etc. an seine Bedürfnisse anpassen. (c) Jeder Benutzer kann jederzeit neue Funktionen hinzufügen. Diese Funktionen können sofort integriert und getestet werden, ohne den Editor neu starten zu müssen [Blandy 2009], dies erleichtert die Implementierung neuer Funktionen. (d) Allgemeine Funktionen, wie das automatische Einrücken von Programmcode entsprechend den syntaktischen Strukturen (engl. *indentation*), berücksichtigen die jeweiligen Strukturen und Regeln der Sprache, auf die sie angewandt werden [Blandy 2009, S. 267f]. Die *editing units* [Khwaja und Urban 1993] sind also sprachabhängig: «EMACS can be programmed to understand the Syntax of the language being edited and provide operations particular to it.» [Stallman 1981, S. 147] (e) Jede Funktion kann über ihren Funktionsnamen oder über ein Tastaturkürzel aufgerufen werden – beide sind ebenfalls von jedem Benutzer änderbar.

Wir verwenden *XEmacs* in der Version 21.4.22 «Instant Classic». *XEmacs* und *Emacs* unterscheiden sich hinsichtlich einiger Implementierungsdetails. Da jeder frei ist, *Emacs* zu modifizieren, können spezielle Modifizierungen sich in einem grösseren Benutzerkreis durchsetzen, eine solche Variante ist *XEmacs*. Für verschiedene Varianten spielen zudem Lizenzmodelle eine Rolle, beide Versionen – *GNU Emacs* und *XEmacs* – und deren Quelltexte sind jedoch frei zugänglich.[2]

Die meisten der in Abschnitt 4.1.1 angesprochenen Texteditoren beziehen sich ausdrücklich auf *Emacs* als Inspirationsquelle. Viele der in Abschnitt 3.4 vorgestellten Schreibhilfen verwenden *Emacs* als Editor und implementieren spezifische Funktionen. *Emacs* bietet sich hierfür an, weil die Erweiterbarkeit eines der Designprinzipien ist und verschiedene Funktionen in *Modi* gebündelt werden können, sodass Funktionen für eine bestimmte Benutzergruppe gezielt verfügbar gemacht werden können. Zudem ist *Emacs* plattformunabhängig.

Ursprünglich wurde *Emacs* als Texteditor für Programmierer entwickelt. Jedoch verwenden auch professionelle Autoren natürlichsprachlicher Texte *Emacs*,

1. Wir verwenden die Schreibweise *Emacs*, in der Literatur sind verschiedene andere Schreibweisen zu finden, etwa *EMACS* oder *emacs*.
2. Für eine Diskussion der Lizenzmodelle verweisen wir auf Stephen J. Turnbull, «XEmacs vs. GNU Emacs», 22.12.2009, online, `http://www.xemacs.org/About/XEmacsVsGNUemacs.html` (zuletzt besucht am 8.12.2010, 19:34).

etwa Wissenschaftler verschiedener Disziplinen[3]. Für die Erstellung von Texten in LaTeX, *troff* oder anderen Auszeichnungssprachen bietet *Emacs* spezifische Unterstützung bei der Verwendung der Auszeichnungen oder zum Kompilieren. Oft stellen Verlage LaTeX-Vorlagen zur Verfügung, um Autoren zu erleichtern, die typographischen Vorgaben einzuhalten – die Verwendung von MS *Word* wird in der Regel missbilligt.

Emacs (und seine Varianten) ist ein *professionelles* Schreibwerkzeug, im Gegensatz zu MS *Word* oder dessen Open-Source-Abklatsch *OpenOffice*, so schreibt Stephenson, dessen Erfahrungen mit MS *Word* wir in Abschnitt 5.3 bereits zitiert haben:

> I use emacs, which might be thought of as a thermonuclear word processor. It was created by Richard Stallman; enough said. It is written in Lisp, which is the only computer language that is beautiful. It is colossal, and yet it only edits straight ASCII text files, which is to say, no fonts, no boldface, no underlining. In other words, the engineer-hours that, in the case of Microsoft Word, were devoted to features like mail merge, and the ability to embed feature-length motion pictures in corporate memoranda, were, in the case of emacs, focused with maniacal intensity on the deceptively simple-seeming problem of editing text. If you are a professional writer—i.e., if someone else is getting paid to worry about how your words are formatted and printed—emacs outshines all other editing software in approximately the same way that the noonday sun does the stars. It is not just bigger and brighter; it simply makes everything else vanish. [Stephenson 1999, S. 95]

Wir verwenden *XEmacs*, um die von uns vorgeschlagenen Funktionen prototypisch zu implementieren. Die Implementierung ist in zweifacher Hinsicht «prototypisch»: (a) Unsere Implementierung ist prototypisch, da wir nicht alle der in Kapitel 4 vorgeschlagenen Funktionen implementieren, sondern nur repräsentative Vertreter der einzelnen Funktionsklassen. Aus diesen Implementierungen lassen sich weitere leicht ableiten. (b) Unsere Implementierung zeigt, dass computerlinguistische Ressourcen sich prinzipiell in bestehende Editoren integrieren lassen, um in linguistisch motivierten Funktionen verwendet zu werden. Im Gegensatz zu üblichen Textbearbeitungsprogrammen erlaubt es *XEmacs*, neue Funktionen hinzuzufügen, die sich dann wie die bereits bestehenden verhalten.

Wie schon in Abschnitt 4.4 angesprochen, sollten sich alle Funktionen eines Editors ähnlich verhalten. Die von uns vorgeschlagenen linguistisch unterstützten Funktionen gehen ganz klar davon aus, dass der Benutzer (der Autor) die Kontrolle über den Text hat. Daher können solche Funktionen nicht in einen Editor integriert werden, der ein ganz anderes Bedienermodell verfolgt, wie etwa MS *Word*, über das Lanier urteilt:

> You might have had the experience of having Microsoft Word suddenly determine, at the wrong moment, that you are creating

3. Journalisten sind ebenfalls professionelle Autoren, denen das jeweilige Verlagshaus jedoch spezielle Redaktionssysteme zur Verfügung stellt.

> an indented outline. […] From my point of view, this type of design features is nonsense, since you end up having to work more than you would otherwise in order to manipulate the software's expectations of you. The real function of the feature isn't to make life easier for people. Instead, it promotes a new philosophy: that the computer is evolvoing into a life-form that can understand people better than people can understand themselves. [Lanier 2010, S. 20]

XEmacs verfolgt dagegen das wesentlich sinnvollere Prinzip, wie es das erste einleitende Zitat diese Kapitels beschreibt: «It is very important that the user feel in control of the editor and not the other way around.» [Wood 1981] Zudem ist *XEmacs* frei verfügbar, die von uns entwickelten Funktionen können also interessierten Autoren leicht zur Verfügung gestellt werden. Diese Implementierung belegt, dass eine Ergänzung eines Editors um zusätzliche Funktionen unter Einbindung computerlinguistischer Ressourcen prinzipiell möglich ist.

Wie Hausdorf [2005] zeigt, sind Autoren bereit, Arbeitsschritte, Funktionen und Abläufe zu erlernen, wenn diese Lösungen ermöglichen, die schon immer vermisst wurden und mit bisherigen Werkzeugen aus verschiedenen Gründen nicht umgesetzt werden konnten. Er folgert aus seinen Evaluationsdaten für Benutzer des von ihm entwickelten *ScientiFix*:

> Arbeitsprozesse werden bewußt überdacht und alternative werkzeugseitige Umsetzungen ausprobiert. […] der Textproduzent [muss – C.M.] die Möglichkeiten und Funktionen seines Werkzeugs erst kennen und beherrschen […], bevor er die Freiheit besitzt, einen für sich optimalen Arbeitsprozeß in der Werkzeugumgebung abzubilden. [Hausdorf 2005, S. 217]

Aufgrund seiner Implementierungsprinzipien kann *XEmacs* von jedem Benutzer entsprechend seinen Bedürfnissen angepasst werden – dies betrifft sowohl die verfügbaren Funktionen, deren Aufruf und Wirkungsweise als auch die Bedienoberfläche des Editors. Zur Bedienoberfläche zählt neben der Menüleiste auch die eigentliche Arbeitsfläche. *XEmacs* ist also optimal geeignet, die von Hausdorf beobachteten Arbeitsweisen zu unterstützen.

Im Folgenden zeigen wir bereits vorhandene Funktionen in *XEmacs*, auf die wir für die Implementierung zurückgreifen. Dies sind einerseits Funktionen, die für verschiedene formale Sprachen zur Verfügung stehen und die wir auch für natürliche Sprachen verwenden können (Abschnitt 7.1.1). Andererseits können wir bestehende Funktionen kombinieren, um bereits einen Teil der in Abschnitt 4.3 aufgezeigten Funktionen zu implementieren, die keine linguistischen Ressourcen benötigen (Abschnitt 7.1.2).

7.1.1 Bereits vorhandene Unterstützung im *XEmacs*

XEmacs bietet Funktionen für eine Vielzahl von Programmier- und Auszeichnungssprachen. Einige dieser Funktionen können auch für die Bearbeitung natürlichsprachlicher Texte verwendet werden. Hier stellen wir Funktionen und Prinzipien von *XEmacs* vor, die in der aktuellen Standardversion bereits

implementiert sind. Die Dokumentation von Funktionen kann in *XEmacs* selbst mit C-h f oder M-x describe-function, die Definition von Variablen kann mit C-h v aufgerufen werden.

7.1.1.1 Identifzieren von Strukturen

Bereits vorhanden in *XEmacs* sind Funktionalitäten zum Identifizieren von Wörtern und Sätzen. Wir verwenden jeweils eine pragmatische Definition, die die grundlegendsten Eigenschaften der betrachteten Sprache – Deutsch – berücksichtigt und die benötigten Informationen für die beabsichtigte Verarbeitung bereitstellt.

Satz Im Deutschen beginnt ein Satz mit einem Grossbuchstaben, d. h., die erste Wortform eines Satzes wird unabhängig von ihrer Wortart mit einer Majuskel am Anfang geschrieben. Das Ende eines Satzes wird mit einem Interpunktionszeichen gekennzeichnet, das entweder ein Satzpunkt, ein Fragezeichen oder ein Ausrufezeichen ist. Dem ersten Buchstaben eines Satzes geht entweder ein Leerzeichen voraus, ein Zeilenumbruch, ein beliebig grosser Leerraum (erzeugt durch Leerzeichen, Tabulatoren, Zeilenumbrüche in beliebiger Kombination) oder er ist das erste Zeichen eines Textes. Dem Satzendzeichen folgt entweder ein Leerzeichen, ein Zeilenumbruch, ein beliebig grosser Leerraum (erzeugt durch Leerzeichen, Tabulatoren, Zeilenumbrüche in beliebiger Kombination) oder es ist das letzte Zeichen eines Textes. Abstrakt geht einem Satzanfang jeweils ein Satzende voraus (ausser dem ersten Satz eines Textes) und einem Satzende folgt ein Satzanfang (ausser dem letzten Satz). Es genügt daher, das Satzende zu erkennen.

Für unsere prototypische Implementierung verwenden wir die bereits in *XEmacs* vorhandene Satzerkennung, basierend auf der Anwendung eines regulären Ausdrucks, wie in der Dokumentation der Variablen `sentence-end` beschrieben:

> In order to be recognized as the end of a sentence, the ending period, question mark, or exclamation point must be followed by two spaces, unless it's inside some sort of quotes or parenthesis.
>
> All paragraph boundaries also end sentences, regardless.[4]

Diese Definition kann – wie alle Variablen und Funktionen in *XEmacs* – angepasst werden. Die Erkennung von Anfang und Ende eines Satzes ist eine scheinbar triviale Aufgabe, die für die meisten der vorgeschlagenen und bereits umgesetzten Funktionen essenziell ist. Wir halten uns hier an die in Abschnitt 4.4 dargelegte Forderung nach der Verwendung möglichst «einfacher» Ressourcen, um das Zusammenspiel verschiedener Ressourcen nicht unnötig zu verkomplizieren und fundierte Aussagen über erwartete Resultate machen zu können.

4. Definition der Variablen `sentence-end`, siehe http://www.xemacs.org/Documentation/beta/html/lispref_45.html#IDX2011 (zuletzt besucht am 8.12.2010, 19:34) .

Wort Im Deutschen beginnt ein Wort mit einem Buchstaben und endet mit einem Buchstaben. Der erste Buchstabe muss in bestimmten Konstellationen (abhängig von der Stellung im Satz und abhängig von der Wortart) eine Majuskel sein. Vor und nach einem Wort können Leerräume verschiedener Art und Länge stehen. Nach einem Wort können zudem Interpunktionszeichen folgen. Es gibt jedoch auch eine einfachere Beschreibung: Ein Wort ist jede Kombination von Zeichen, die ausschliesslich aus Buchstaben besteht. Zudem sollte eine solche Buchstabenkette wohlgeformt sein, d. h. tatsächlich eine deutsche Wortform sein – dies könnte durch eine morphologische Analyse geprüft werden. Da es uns hier nicht um Rechtschreibkorrektur geht und Autoren natürlich jederzeit frei sind, neue Wörter zu «erfinden», können wir uns auf eine Definition beschränken, die eine Wortform mit einer Zeichenkette gleichsetzt, die lediglich aus Buchstaben besteht, unabhängig davon, ob es sich dabei um Majuskeln oder Minuskeln handelt. Dies entspricht der Verwendung des regulären Ausdrucks `\w`[5]. Komposita mit Bindestrich, wie etwa «*Cursor-Position*», werden durch diesen regulären Ausdruck nicht als *eine* Wortform erkannt. Hier muss also eine Anpassung erfolgen, soll ein solches Kompositum als eine Wortform behandelt werden.

In *XEmacs* werden die Möglichkeiten der Erkennung von Strukturen (wie etwa Wortform und Satz) ausgenutzt, um dem Benutzer bereits vordefinierte Funktionen anzubieten, die diese Strukturen verwenden. Wir gehen im Folgenden jeweils nur auf die Funktionen ein, die für die Bearbeitung natürlichsprachlicher Texte sinnvoll sind. Funktionen, die sich auf programmiersprachliche Konstrukte beziehen – z. B. S-Expressions –, erwähnen wir nicht, ebenso berücksichtigen wir nicht Funktionen, die sich auf sehr abstrakte Einheiten beziehen, wie Zeilen oder Regionen[6].

7.1.1.2 Syntaxhighlighting (Informationsfunktionen)

In Programmiersprachenmodi ist das Hervorheben von Schlüsselwörtern und syntaktischen Strukturen essenziell. Mittels regulärer Ausdrücke werden Strukturen identifiziert, denen *text properties* zugewiesen werden. Diese *text properties* werden verwendet, um die entsprechenden Zeichen hervorzuheben:

> Lisp code can attach text properties to the characters in a buffer. A text property has a name and a value, and both can be arbitrary Lisp objects. […] Text properties can specify how Emacs should display the text, and also how Emacs should respond to mouse gestures applied to it. Text properties can even specify special keyboard commands available to the user only when the cursor is on that text. [Blandy 2009, S. 268][7]

5. Siehe http://www.xemacs.org/Documentation/21.5/html/lispref_46.html#SEC613 (zuletzt besucht am 8.12.2010, 19:34).
6. «The text between point and the mark is known as the region.» http://www.xemacs.org/Documentation/beta/html/lispref_43.html#SEC553 (zuletzt besucht am 8.12.2010, 19:34).
7. Siehe auch: «Text properties are an alternative interface to extents […], and are built on top of them.» http://www.xemacs.org/Documentation/beta/html/lispref_44.html#SEC583 (zuletzt besucht am 8.12.2010, 19:34)., «An extent is a region of text (a start position and an end position) that is displayed in a particular face and can have certain other properties […] Extents can overlap each other.», http://www.xemacs.org/Documentation/beta/html/lispref_48.html#SEC627 (zuletzt besucht am 8.12.2010, 19:34).

Auf diese *text properties* kann von weiteren Funktionen zugegriffen werden, um den Cursor zu bestimmten Elementen zu bewegen oder die zugehörige Zeichenkette zu manipulieren.

Für natürliche Sprachen existieren hingegen keine solche Funktionen. In Ansätzen finden wir Hervorhebungen und Markierungen von natürlichsprachlichen Elementen, wenn diese als Strukturelemente, beispielsweise in LaTeX-Dokumenten, ausgezeichnet sind – hier wird jedoch lediglich über reguläre Ausdrücke auf Auszeichnungselemente zugegriffen – also auf die Strukturen der verwendeten Auszeichnungssprache.

7.1.1.3 Bewegungsfunktionen

Es ist möglich, den Cursor wortweise zu bewegen mittels `forward-word` (M-f) und `backward-word` (M-b). Analog kann der Cursor mittels `forward-sentence` (M-e) und `backward-sentence` (M-a) auch satzweise bewegt werden.

7.1.1.4 Modifikationsfunktionen

XEmacs bietet verschiedene vordefinierte Funktionen, um einen Text zu manipulieren. Dies sind (a) verschiedene Varianten von Vertauschungsfunktionen, (b) Funktionen, die Gross- und Kleinschreibung verändern, (c) Funktionen zum Löschen von Wortformen oder Sätzen und (d) Funktionen zum Ersetzen von Wortformen unter Berücksichtigung der Gross-/Kleinschreibung.

Vertauschen von Elementen Vertauscht werden können zwei Zeichen, zwei Wortformen und zwei Sätze. Die Funktionen `transpose-chars` (C-t) und `transpose-preceding-chars` vertauscht zwei benachbarte Zeichen, relevant ist die aktuelle Position des Cursors. Mit der Funktion `transpose-words` (M-t) können zwei benachbarte Wortformen vertauscht werden. Die Funktion `transpose-sentences` vertauscht zwei aufeinanderfolgende Sätze.

Gross-/Kleinschreibung Es ist möglich, den ersten Buchstaben einer oder mehrerer Wortformen in eine Majuskel zu wandeln mittels `capitalize-word` (M-c). Diese Funktion ist sehr viel einfacher aufzurufen als die vergleichbare Funktion in *MS Word*, wie in Abschnitt 3.1.2.3 beschrieben. Es können auch alle Buchstaben einer Wortform mittels `upcase-word` und `upcase-region-or-word` (M-u) in Majuskeln gewandelt werden oder mittels `downcase-word` und `downcase-region-or-word` (M-d) alle in Minuskeln.

Löschen von Elementen Die Funktionen `kill-word` (M-d), `backward-kill-word` und `backward-or-forward-kill-word` erlauben das Löschen von Wortformen. Die Funktionen `kill-sentence` (M-k), `backward-kill-sentence` und `backward-or-forward-kill-sentence` erlauben das Löschen von Sätzen.

Ersetzen von Elementen Mit Hilfe der Funktion `query-replace` können beliebig komplexe Elemente ersetzt werden. Berücksichtigt man die Variablen `case-replace` und `case-fold-search`, wird die Gross-/Kleinschreibung des Suchwortes beibehalten. Verwendet man diese Funktion

für das Suchen und Ersetzen von deutschen Wortformen, wird beispielsweise berücksichtigt, ob eine Wortform am Satzanfang steht und darum mit einer Majuskel beginnt.

Für das Ersetzen von deutschen Wörtern ist diese Funktion dagegen nur bedingt geeignet: Wörter der geschlossenen Wortklassen können problemlos ausgetauscht werden, für Wörter der offenen Wortklassen werden verschiedene Wortformen von Such- und Ersatzwort nicht berücksichtigt. Selbst wenn die Funktion `query-replace-regexp` verwendet wird, die reguläre Ausdrücke benutzt, ist dies ein sehr aufwendiges und fehleranfälliges Vorgehen, da die Flexion deutscher Wörter nicht über reguläre Ausdrücke beschrieben werden kann. Es ist also notwendig, für jede Wortformoberfläche des Suchwortes die entsprechende Wortformoberfläche des Ersatzwortes anzugeben, wie schon in Abschnitt 3.1.2.4 beschrieben.

7.1.2 Verwendung vordefinierter Funktionen zum Bearbeiten von natürlichsprachlichen Texten

Bereits in *XEmacs* vorhandene vordefinierte Funktionen verwenden jeweils die beschriebenen Definitionen von Wort und Satz und ermöglichen es, für bestimmte Redigieroperationen – für die keine eigene vordefinierte spezielle Funktion vorhanden ist – eine geringere Anzahl Kernfunktionen zu verwenden, als es mit anderen Textbearbeitungsprogrammen der Fall ist.

Als Beispiel betrachten wir das Vertauschen von Konjunkten, wie bereits im Editierbeispiel für konventionelle Textbearbeitungsprogramme in Abschnitt 3.1.2.2 gezeigt: Um die Variante «*schreiben und lesen*» in «*lesen und schreiben*» zu ändern, lassen sich verschiedene Vorgehen bei verschiedenen Schreibern beobachten. Sehen wir davon ab, die gesamte Konjunktion komplett (oder auch Teile) zu löschen und neu zu schreiben, sondern beschränken uns auf die Verwendung von Funktionen, die die bereits geschriebenen Wortformen umsortieren, kann diese Aktion sehr einfach beschrieben werden: (a) Vertausche «*schreiben*» und «*lesen*» oder (b) Drehe «*schreiben und lesen*» um. Funktion zum «Umdrehen» von Wortgruppen oder «Spiegeln» von Strukturen existieren auch in *XEmacs* nicht. In Abschnitt 7.1.1.4 sind wir jedoch bereits auf Funktionen zum Vertauschen von Elementen eingegangen, die wir hier verwenden können.

Für *MS Word* sind mindestens acht Schritte notwendig, wenn wir Funktionen verwenden, um beispielsweise nicht zeichenweise markieren zu müssen, wie Abbildung 7.1 zeigt.

Schritt	Kommando	Status
		`schreiben und lesen|`
1.	Alt Shift ←	`schreiben und lesen`
2.	Cmd X	`schreiben und |`
3.	Alt ←	`|schreiben und`
4.	Cmd V	`lesen |schreiben und`
5.	Alt Shift →	`lesen schreiben und`
6.	Cmd X	`lesen |und`
7.	Alt →	`lesen und |`
8.	Cmd V	`lesen und schreiben|`

Abbildung 7.1: Kürzestmögliche Sequenz von Funktionen in MS Word*, um «schreiben und lesen» in «lesen und schreiben» zu ändern, unter Verwendung von «Smart Cut and Paste» (Mac Tastenkombinationen; der farbige Hintergrund kennzeichnet die aktuelle Markierung, | kennzeichnet die aktuelle Cursor-Position).*

Abbildung 7.2 zeigt, dass die Verwendung der bereits vorhandenen und oben beschriebenen Funktion `transpose-words` die Anzahl der benötigten Schritte erheblich reduziert – es werden nur drei statt acht Schritten benötigt. Entsprechende Funktionen stehen in MS *Word* nicht zur Verfügung.

Abbildung 7.2: Kürzestmögliche Sequenz von Funktionen in XEmacs*, um «schreiben und lesen» in «lesen und schreiben» zu ändern.*

Schritt	Kommando	Status
		`schreiben und lesen|`
1.	C-u - 2 M-t	`lesen| schreiben und`
2.	M-f	`lesen schreiben| und`
3.	M-t	`lesen und schreiben|`

Wir finden hier eine erste Bestätigung, dass die Möglichkeit, spezifische Einheiten und darauf aufbauende Funktionen zu verwenden, kürzere Sequenzen von Funktionen zur Umsetzung einer Redigierabsicht zur Folge hat. Dies reduziert die Gefahr, einen Schritt zu vergessen oder eine ungünstige Reihenfolge zu wählen, die die Anzahl der benötigten Funktionen absolut noch erhöht. Wir finden hier auch eine Bestätigung dafür, dass linguistisch motivierte Funktionen nicht notwendigerweise den Einbezug linguistischer Ressourcen bedingen. Die Ausnutzung der Eigenschaften einer natürlichen Sprache erlaubt es bereits, die Anzahl der benötigten Funktionen und die Komplexität der Sequenz dieser Funktionen zu reduzieren.

Eine weitere bereits angesprochene Redigieroperation ist die Änderung der Gross- und Kleinschreibung (siehe Abschnitt 3.1.2.3). Die Funktion zum Ändern des ersten Buchstabens eines Wortes hinsichtlich Gross- und Kleinschreibung kann, wie alle anderen, direkt über die Tastatur aufgerufen werden als `capitalize-word` (M-c). Es muss nicht wie in MS *Word* erst zu einer im Menü versteckten Funktion navigiert werden. Vor allem ist es nicht notwendig, dazu eine Hand von der Tastatur zu lösen, um die Maus für den Aufruf dieser Funktion zu verwenden.

Schon Stallman [1981] spricht von Funktionen explizit für natürliche Sprachen:

> For editing English text, commands have been written to move the cursor by words, sentences, and paragraphs, and to delete them; to fill and justify paragraphs; and to move blocks of text to the left or to the right. Other commands convert single words or whole regions to upper or lower case. There are also commands which manipulate the command string for text justifier programs: some insert or delete underlying commands, and other insert or delete font-change commands. [Stallman 1981, S. 148]

Die Auflistung dieser Funktionen ist jedoch unsystematisch, einige beziehen sich auf linguistische Einheiten, etwa das wortweise Bewegen des Cursors, andere beziehen sich auf die Ausrichtung des Texts innerhalb eines Fensters, wieder andere auf Parameter von Funktionen.

Im Sommer 2001 erscheint im Emacs-Wiki der Vorschlag für einen *Natural Language Mode*[8]; im Sommer 2010 umfasst die Liste der vorgeschlagenen Funktionen unter anderem:

8. «NaturalLanguageMode», Seite im Emacs-Wiki, 28.6.2010, online http://www.emacswiki.org/emacs/?action=browse;id=NaturalLanguageMode (zuletzt besucht am 8.12.2010, 19:34).

- `pluralize-word`: Setzt das Wort an der Cursor-Position in den Plural und funktioniert entsprechend `PluralizeEnglish`.
- `singularize-word`: Setzt das Wort an der Cursor-Position in den Singular.
- `{past,present,future}-tense-verb`: Ändert das Tempus für Verben.
- `change-gender`: Ändert das Genus.

Diese Liste ist zwar auf linguistische Operationen beschränkt, jedoch ebenfalls unsystematisch. Die erwähnte `PluralizeEnglish`-Funktion[9] verwendet `plural.el`[10] von Aaron S. Hawley von 2007 und arbeitet hauptsächlich mit regulären Ausdrücken – was für English schon sehr weit führt – sowie Listen von Ausnahmefällen. Es werden keine computerlinguistischen Ressourcen verwendet. Der Entwickler Charles Sebold kommentiert die Vorschläge zum *Natural Language Mode*:

> My goodness, that's quite ambitious. Wouldn't this require grammar guides in some parsable form, for supported languages? Perhaps Ispell can do some of this now, though—it does recognize some plurals, etc. Not that I think that Emacs can't extend to that point someday. But there are a lot of problems to be overcome before this could become a reality, I should think.[11]

Verfügbare Funktionen zur Bearbeitung natürlichsprachlicher Texte im *XEmacs*, die auf linguistischen Einheiten operieren, existieren also nicht, werden jedoch gewünscht. Eine systematische Konzeption ist bislang nicht erfolgt, die Idee, computerlinguistische Ressourcen einzubinden, wurde bislang nicht umgesetzt.

Einzelne Benutzer implementieren zwar private Ad-hoc-Lösungen, ausgehend von persönlichen Bedürfnissen, etwa das Verschieben des Verbs an das Ende des aktuellen Satzes, das Ändern des Genitivs in den Nominativ, der Wechsel zwischen Partizip II und Infinitiv eines Verbs, das Abtrennen eines Verbzusatzes eines Verbgefüges und Platzieren ans Satzende, wie in `morpho.el` von Yves Forkl[12]. Hier werden jedoch keine computerlinguistischen Ressourcen verwendet, sondern ebenfalls formale Eigenschaften des Deutschen ausgenutzt; bei der Verschiebung des Verbs an das Satzende wird davon ausgegangen, dass das Wort an der aktuellen Cursor-Position ein Verb ist, es wird nicht nachgeprüft, ob dies zutrifft. Ebenso wird beim Abtrennen des Verbzusatzes eines Verbgefüges die aktuelle Cursor-Position als Trennstelle verwendet. Die Resultate dieser Funktionen sind zufriedenstellend, jedoch beruht die Implementierung rein auf der Oberfläche von Zeichenketten und auf mit regulären Ausdrücken beschreibbaren Vorgängen, die *editing units* sind nicht durch explizit formuliertes linguistisches Wissen definiert.

9. «PluralizeEnglish», Seite im Emacs-Wiki, 5.9.2008, online http://www.emacswiki.org/emacs/PluralizeEnglish (zuletzt besucht am 8.12.2010, 19:34).
10. http://www.emacswiki.org/emacs/plural.el (zuletzt besucht am 8.12.2010, 19:34).
11. «NaturalLanguageMode», 28.6.2010, online http://www.emacswiki.org/emacs/?action=browse;id=NaturalLanguageMode (zuletzt besucht am 8.12.2010, 19:34).
12. Zur Verfügung gestellt am 5. Juni 2007, Message-ID <4665C011.7030808@arcor.de>.

7.1.3 Zusammenfassung: *XEmacs* als Editor

Wir verwenden für die Implementierung linguistisch unterstützter Funktionen *XEmacs*. Einerseits existieren bereits verschiedene Konzepte und Funktionen, die wir für unsere Implementierung nutzen können, andererseits ermöglicht es *XEmacs* auf sehr komfortable Weise, neue Funktionen hinzuzufügen. Im Gegensatz zu anderen Textbearbeitungsprogrammen verwendet *XEmacs* über einzelne Zeichen hinausgehende Funktionen bereits in Standardfunktionen. *XEmacs* hat ein Bedienkonzept, das mit den Designprinzipien der von uns vorgeschlagenen linguistisch basierten Funktionen kompatibel ist: Der Benutzer kontrolliert den Editor, nicht umgekehrt. Benutzer können *XEmacs* entsprechend persönlichen Vorlieben anpassen, für andere Editoren muss sich der Benutzer dem Programm anpassen.

Bei *XEmacs* handelt es sich um ein *Open-Source*-Produkt und es existiert eine aktive Entwicklergemeinde. Einzelne Personen implementieren persönlich motivierte Funktionen und stellen sie für den allgemeinen Gebrauch zur Verfügung. Es existieren Konventionen zur Benennung und zum Aufruf solcher neuen Funktionen, sodass neue Funktionen einfach zu erlernen sind [Blandy 2009, S. 274]. Wir konnten ebenfalls feststellen, dass es bereits verschiedene Funktionen gibt, die es in Ansätzen erlauben, linguistische Elemente zu modifizieren. Diese Funktionen unterstützen jedoch einerseits andere Sprachen – in der Regel Englisch – und andererseits beruhen sie nicht auf einem fundierten Konzept, sondern befriedigen ad hoc auftretende Bedürfnisse.

Wir zeigen im Folgenden, wie unter Einsatz der in Kapitel 6 ausgewählten Ressourcen systematisch die in Abschnitt 4.2 entworfenen Funktionen implementiert werden können. Dieses Vorgehen unterscheidet sich von den ad-hoc-Implementierungen einerseits durch die tiefere linguistische Motivierung. Andererseits befriedigen wir nicht konkrete persönliche Bedürfnisse einzelner Benutzer, sondern schätzen aufgrund der Eigenschaften der Ressourcen, der Eigenschaften von entstehenden Texten, häufig auftretender Fehler und Erschwernisse sowie von Rückmeldungen einzelner Benutzer die Sinnhaftigkeit prinzipiell denkbarer Funktionen ab. Wir gehen in Abschnitt 7.3 genauer darauf ein.

7.2 Verwendete Ressourcen

In diesem Abschnitt beschreiben wir die Einbindung der in Kapitel 6 ausgewählten computerlinguistischen Ressourcen in *XEmacs* jeweils anhand einer konkreten Funktion. Diese Funktionen sind bereits direkt von Autoren verwendbar oder können in andere Funktionen eingebunden werden.

7.2.1 Morphologische Analyse und Generierung von Wortformen

Für Funktionen, die die Kategorie einer oder mehrerer Wortformen verwenden, setzen wir *GERTWOL/GERGEN* [Haapalainen und Majorin 1994, Koskenniemi und Haapalainen 1996] ein. Dabei ist *GERTWOL* die Analyse- und *GERGEN* die Generierungskomponente. *GERTWOL* und *GERGEN* stehen uns in einer akademischen Lizenz zur Verfügung, beide sind nicht frei verfüg-

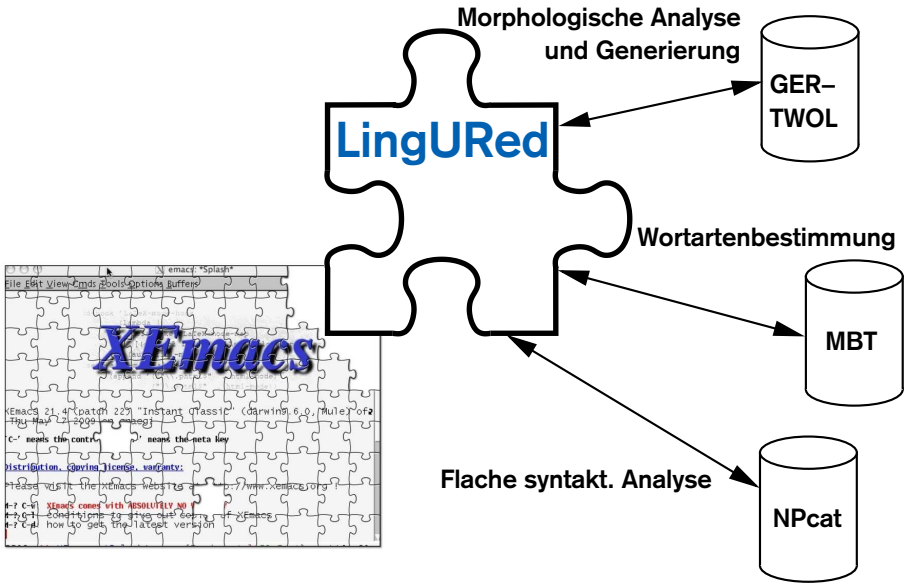


Abbildung 7.3: Einbindung der verwendeten Ressourcen in XEmacs.

bar.[13] Beide werden als *shared library* ausgeliefert, wir verwenden die Version vom 15. Juli 2010, gültig für Deutsch entsprechend den Rechtschreibregeln von 2006 ohne Berücksichtigung der Schweizer Rechtschreibung, jedoch einschliesslich Helvetismen. Um *GERTWOL* und *GERGEN* verwenden zu können, muss jeweils die shared library eingebunden werden. Zur Auswahl von *GERTWOL/GERGEN* verweisen wir auf Abschnitt 6.2.

Da wir die gleiche Schnittstelle sowohl zum Analysieren als auch zum Generieren verwenden, sind wir gezwungen, die «alte» Version, nämlich *LSLING* zu verwenden (siehe die Abschnitte 6.2.4.3 und 6.2.7.3 zur Diskussion von *LSINDEX* und *LSLING*). Beim ersten Aufruf einer Funktion, die *GERTWOL* oder *GERGEN* verwendet, dauert es etwa eine Sekunde, bis das Frontend gestartet ist. Anschliessend läuft dieser Prozess im Hintergrund weiter und weitere Verwendungen liefern das Ergebnis ohne merkliche Verzögerung.[14]

Das Frontend stellt eine Schnittstelle zur *GERTWOL/GERGEN* shared library dar. Theoretisch könnte die shared library direkt in *XEmacs* integriert werden, wir verwenden jedoch einen separaten Prozess via pipe. Über das Frontend kann eine Wortform an *GERTWOL* übergeben werden und eine entsprechende Analyse wird zurückgeliefert. Diese Analyse ist im S-Expression-Format, sodass sie in ELisp direkt ausgewertet werden kann. Für die Generierung werden eine Grundform und eine *LSLING*-Kategorie übergeben, das Frontend liefert als S-Expression die generierten Wortformen zurück.

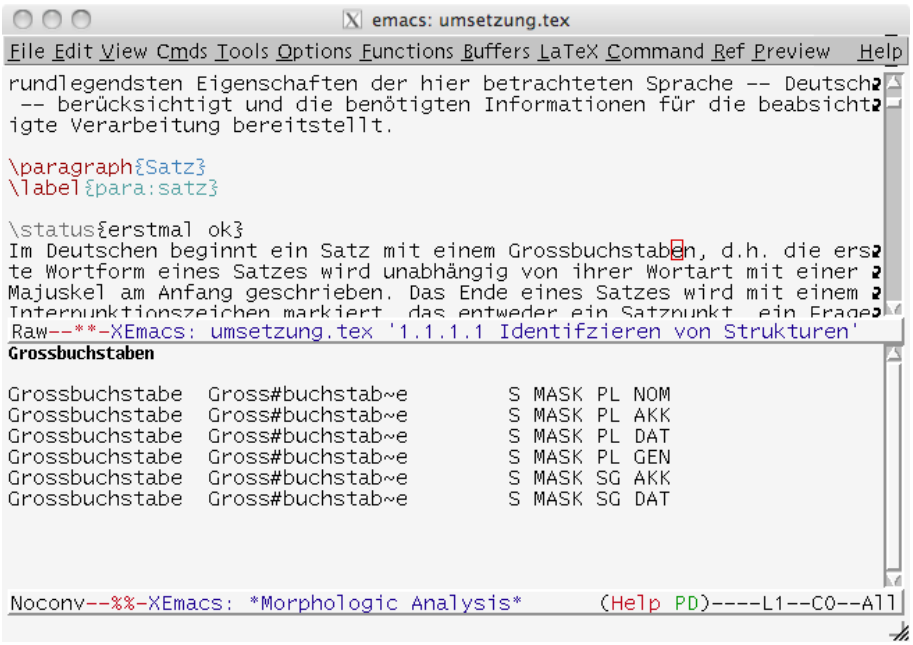
Als Referenzimplementierung zeigen wir die Möglichkeit, die Wortform an der aktuellen Cursor-Position analysieren zu lassen. Listing 7.1 auf der nächsten Seite zeigt den Pseudocode für diese Funktion, Abbildung 7.4 auf der nächsten Seite zeigt die Anwendung der Funktion. Der Cursor ist auf «*Grossbuchstaben*» platziert. Betrachtet man diese Wortform isoliert – d. h. ohne Berücksichtigung des syntaktischen Kontextes –, ist sie stark ambig.

13. Das bedeutet, dass für Funktionen, die hier *GERTWOL/GERGEN* als Ressource verwenden, unter Umständen andere Ressourcen eingebunden werden müssen, wenn eine entsprechende Lizenz nicht zur Verfügung steht.
14. Natürlich kann *XEmacs* auch so konfiguriert werden, dass entweder immer beim Starten von *XEmacs* oder beim Laden bestimmter Dokumente *GERTWOL/GERGEN* gestartet werden.

Listing 7.1: Pseudocode für die Einbindung von GERTWOL/GERGEN *zur Analyse (*`get-analysis`*).*

```
get word at point
ifnot GERTWOL running
  start GERTWOL
give word to GERTWOL
get analysis
show analysis in new buffer
```

Abbildung 7.4: Einbindung von GERTWOL *in* XEmacs*: Analyse von Wortformen.*



Als zweites Beispiel zeigen wir die Änderung des Numerus: Handelt es sich um ein Substantiv im Singular, wird die Pluralform erzeugt – ist die Kategorie der ursprünglichen Wortform ambig, wird eine Liste von Wortformen zurückgeliefert, aus der der Benutzer auswählen kann – und die ursprüngliche Singularform wird durch die Pluralform ersetzt. Handelt es sich um ein Substantiv im Plural, wird die Singularform erzeugt und im Text ersetzt.

Listing 7.2: Pseudocode für die Einbindung von GERTWOL/GERGEN *zur Änderung des Numerus.*

```
get-analysis
foreach reading
  toggle number
  generate wordform
get unique list of wordforms
show in buffer for selection
replace word at point with selected wordform
```

Listing 7.2 zeigt den Pseudocode, Abbildung 7.5 auf der nächsten Seite zeigt die Anwendung der Funktion. Wir positionieren den Cursor wiederum auf «*Grossbuchstaben*» und rufen die entsprechende Funktion auf. Die Wortform wird farblich hervorgehoben[15] – wir verwenden hier die in Abschnitt 7.1.1.2 auf Seite 192 vorgestellten *extents* und *text properties*. Für die Plurallesarten (siehe Abbildung 7.4) wird jeweils die Singularform generiert, für die Singularlesart wird die korrespondierende Pluralform generiert. Daraus ergeben sich drei verschiedene Oberflächen, die der Benutzer durch die Eingabe der entsprechenden Ziffer auswählt. Die Optionen für die erwartete Eingabe sind, wie für andere Funktionen ebenfalls üblich, in eckigen Klammern angegeben. Soll

15. Die Farbgebung kann wie üblich im *XEmacs* über den Menüpunkt `Options` angepasst werden.

nichts ausgewählt – und damit nichts geändert werden –, wird die Funktion wie üblich durch Eingabe von q beendet. In beiden Fällen wird die grüne Hervorhebung wieder entfernt und der Benutzer kann weiter wie gewohnt schreiben oder andere Funktionen aufrufen.

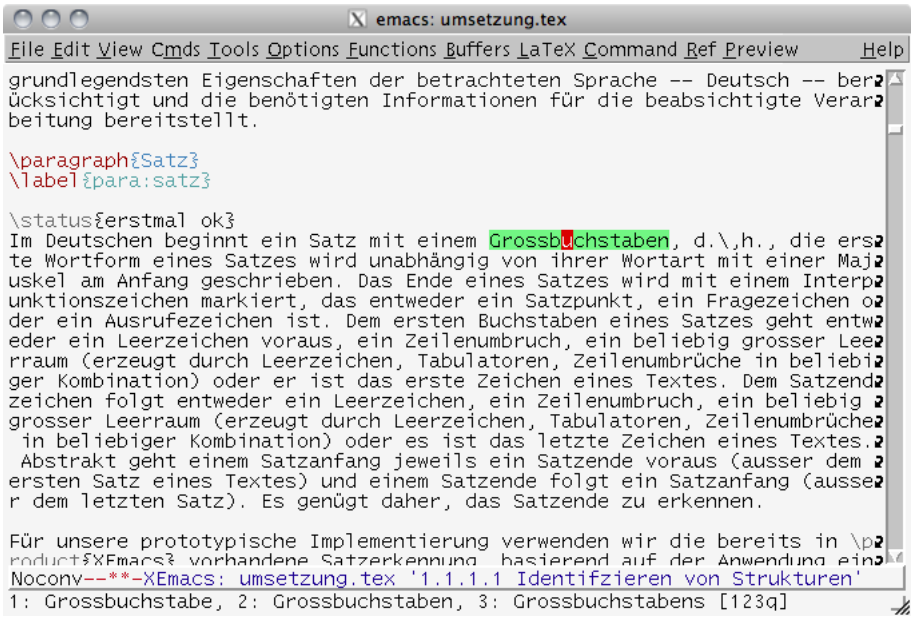


Abbildung 7.5: Einbindung von GERTWOL/GERGEN *in* XEmacs*: Ändern des Numerus.*

7.2.2 Bestimmung von Wortarten für einzelne Wortformen

Für die Bestimmung von Wortarten für einzelne Wortformen verwenden wir *Mbt* (Memory-based tagger generation and tagging) [Daelemans et al. 1996, 2010, Zavrel und Daelemans 1999] in der Version 3.1.3, entwickelt von der *Induction of Linguistic Knowledge Research Group* an der Universität Tilburg.[16] *Mbt* ist unter einer GNU-GPL-Lizenz frei verfügbar. Zum Trainieren des Wortartenbestimmers verwenden wir *Tüba-D/Z* (Tübinger Baumbank des Deutschen) [Telljohann et al. 2009]. Die in *Tüba-D/Z* verwendeten Annotierungen entsprechen dem *STTS* (Stuttgart-Tübingen Tagset)[17]. Zur Auswahl von *Mbt* und *Tüba-D/Z* verweisen wir auf Abschnitt 6.3.

Für die Einbindung von *Mbt* in *XEmacs* haben wir eine Schnittstelle entwickelt, die ebenfalls als separater Prozess funktioniert. Diese erlaubt es, für einen Satz[18] die Wortarten der einzelnen Wortformen[19] bestimmen zu lassen und die entsprechenden Wortartenmarkierungen als *text properties* den einzelnen Wortformen hinzuzufügen. Auf diese *text properties* kann dann von weiteren Funktionen zugegriffen werden, um bestimmte Elemente zu markieren, den Cursor dorthin zu bewegen oder das zugehörige Wort zu manipulieren. Als Beispiel zeigen wir in Listing 7.3 auf der nächsten Seite den mit *Mbt* annotierten Satz:

16. http://ilk.uvt.nl/mbt/ (zuletzt besucht am 8.12.2010, 19:34).
17. http://www.sfs.uni-tuebingen.de/Elwis/stts/stts.html (zuletzt besucht am 8.12.2010, 19:34).
18. Wir verwenden hier *Satz* in der Definition von *XEmacs*, siehe Abschnitt 7.1.1.1 auf Seite 191.
19. Wir verwenden dafür *Wort* in der Definition von *XEmacs*, siehe Abschnitt 7.1.1.1 auf Seite 192.

(7.15) Wegen der Inflationsgefahr soll die stark gestiegene Kreditvergabe der Banken leicht gedrosselt werden.[20]

Listing 7.3: Wortartenmarkierungen für den Beispielsatz 7.15.

```
Wegen/APPR der/ART Inflationsgefahr//NN soll/VMFIN die/ART
stark/ADJD gestiegene/ADJA Kreditvergabe/NN der/ART
Banken/NN leicht/ADJD gedrosselt//VVPP werden/VAINF ./\$.
```

Jeder Wortform folgt nach einem Schrägstrich (/) das Tag für die ermittelte Wortform. Wenn doppelte Schrägstriche (//) auftreten, handelt es sich um eine Hypothese. Wir berücksichtigen diese Information nicht und verwenden lediglich das Tag selbst. Die Qualität der so ermittelten Wortarten hängt von der Qualität der Annotation des Trainingstextes (hier also von *Tüba-D/Z*) ab.

Listing 7.4: Pseudocode für die Einbindung von Mbt *in* XEmacs *(*`pos-tag-sentence`*).*

```
get sentence at point
tokenize sentence
ifnot MBT running
  start MBT
send tokenized sentence to MBT
get results as pairs of word and pos
foreach word
  attach pos as text property
```

Listing 7.5: Pseudocode für die Verwendung von Mbt *zur Hervorhebung von finiten Verben (*`show-finverb`*).*

```
pos-tag-sentence
foreach word
  if POS == VVFIN of VMFIN or VAFIN
     set text property background color
```

Listing 7.4 zeigt den Pseudocode für die Einbindung von *Mbt* in *XEmacs*. Listing 7.5 zeigt den Pseudocode für eine konkrete Verwendung dieser Einbindung: Finite Verben – also Wortformen, die von *Mbt* als `VVFIN` markiert werden – werden hervorgehoben. Dazu wird dem entsprechenden Wort ein *text property* hinzugefügt.

Abbildung 7.6 auf der nächsten Seite zeigt die Verwendung von *Mbt* zur Auszeichnung finiter Verbformen. Der Cursor ist im dritten behandelten Satz positioniert, mit `VVFIN` markierte Wortformen haben einen rosa Hintergrund erhalten. Wird nach dem Aufruf der Funktion eine beliebige andere Taste gedrückt – entweder zum Weiterschreiben oder zum Aufruf einer anderen Funktion –, verschwinden die farbigen Hervorhebungen wieder. Es kann auch der gesamte Text behandelt werden. Beim ersten Aufruf der Funktion dauert es etwa eine Sekunde, bis *Mbt* gestartet ist. Der Prozess läuft anschliessend im Hintergrund weiter und die nächsten Aufrufe der Funktion zeigen das Resultat ohne Verzögerung.[21]

7.2.3 Bestimmung von Substantivgruppen

Für einige Funktionen ist es notwendig, nicht einzelne Wortformen, sondern Wortgruppen zu erkennen und zu kategorisieren, um sie hervorheben oder manipulieren zu können. Zur Erkennung von Substantivgruppen verwenden

20. Text aus der Neuen Zürcher Zeitung, 14. März 2010, «Wen warnt vor neuer Krise», 08:39, NZZ Online http://www.nzz.ch/nachrichten/international/peking_warnt_vor_rueckfall_in_die_krise_1.5210482.html (zuletzt besucht am 8.12.2010, 19:30)

21. Auch *Mbt* könnte direkt beim Starten von *XEmacs* aufgerufen werden, sodass keine Verzögerung bei der ersten Verwendung auftritt.

Abbildung 7.6: Einbindung von Mbt *in* XEmacs*: Hervorhebung finiter Verben.*

wir unsere in Abschnitt 6.4.3 vorgestellte Ressource *NPcat*. *NPcat* benutzt die bereits präsentierten Ressourcen *GERTWOL* und *Mbt*. Zur Diskussion der Notwendigkeit einer eigenen Entwicklung verweisen wir auf Abschnitt 6.4.

Als Substantivgruppe behandeln wir Wortgruppen aus mindestens zwei Wortformen, die aus einer Folge von einem optionalen Artikel, optionalen Adjektiven und einem Substantiv bestehen. Adjektive können koordiniert werden (durch «*und*» oder «*oder*»), ebenso können Adverben oder adverbial gebrauchte Adjektive enthalten sein. Die Substantivgruppe enthält nur *ein* Substantiv und wird nicht von anderen Wortgruppen unterbrochen. Das zugrundeliegende Kriterium zur Selektion von Wortformen, die zu einer Substantivgruppe gehören, ist die Verwendung als Objekt für Informations- und Bewegungsfunktionen sowie Modifikationsfunktionen. Für Bewegungsfunktionen und Modifkationsfunktionen müssen Konjunktionen und Adverben innerhalb einer Substantivgruppe zu dieser hinzugezählt werden, um etwa die gesamte Substantivgruppe zu verschieben. Für Modifikationsfunktionen, die die Substantivgruppe verändern, sind lediglich Artikel, Adjektiv(e) und das Substantiv betroffen.

Listing 7.6: Substantivgruppen und deren Kategorien aus dem Beispielsatz 7.15.

```
der/ART Inflationsgefahr/NN => FEM SG GEN, FEM SG DAT
die/ART stark/ADJD gestiegene/ADJA Kreditvergabe/NN => FEM SG ACC, FEM SG NOM
der/ART Banken/NN => FEM PL GEN
```

Aus dem Beispielsatz 7.15 auf Seite 200 lassen sich die in Listing 7.6 dargestellten Substantivgruppen und deren Kategorien ermitteln. Im Fall von «*die stark gestiegene Kreditvergabe*» gehört «*stark*» nach unseren Kriterien nicht zur Substantivgruppe, die entsprechende Analyse wird auch nicht zur Ermittlung der Kategorie der Substantivgruppe verwendet. Wir schliessen adverbial gebrauchte Adjektive wie «*stark*» (und Adverben) jedoch ein, um eine geschlossene Markierung zu ermöglichen. Das gleiche gilt für Substantivgruppen mit Konjunktion wie in «*eine möglichst fehlerfreie und einheitliche Produktion*».

Listing 7.7 auf der nächsten Seite zeigt den Pseudocode für die Einbindung von *NPcat* in *XEmacs*. In Abschnitt 6.4.3 haben wir *NPcat* unter Verwendung

von Perl implementiert, hier verwenden wir ELisp. Die Funktion benutzt die bereits vorgestellte Schnittstellen zu *Mbt* (Listing 7.4 auf Seite 201 für `pos-tag-sentence`).

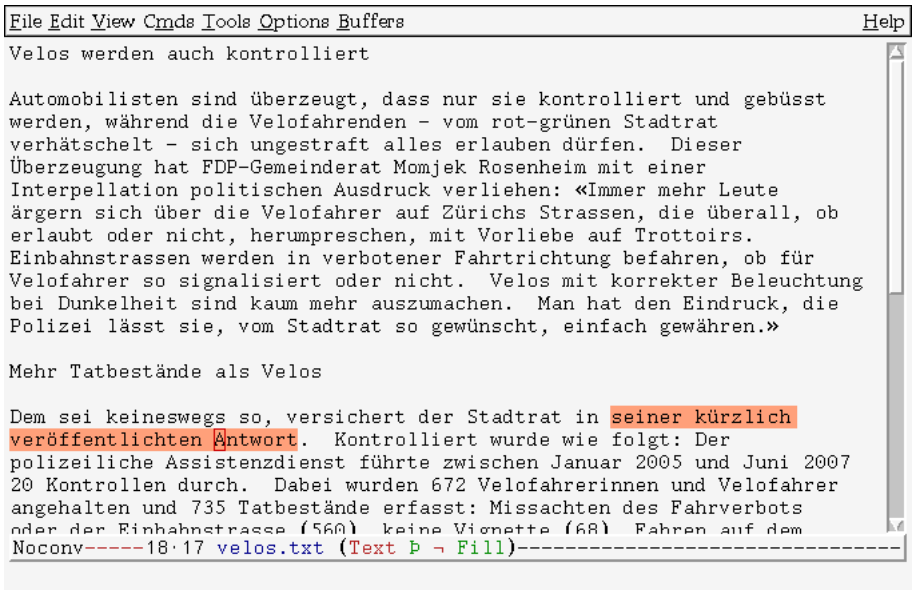
Listing 7.7: Pseudocode für die Einbindung von NPcat *(*`mark-noungroup`*).*

```
pos-tag-sentence
ifnot pos of word at point == NN
  find next NN
get-noungroup-bounds
set region
```

Die Funktion `mark-noungroup` zur Markierung von Substantivgruppen ist analog zu bereits vorhandenen Markierungsfunktionen wie `mark-paragraph` oder `mark-word` implementiert.[22] Von der aktuellen Cursor-Position aus wird nach rechts die nächste Substantivgruppe gesucht und markiert. Zuerst wird für alle Wörter im aktuellen Satz mit `pos-tag-sentence` die Wortart bestimmt. Ist der Cursor auf einem Substantiv platziert, wird dieses verwendet. Ist der Cursor nicht auf einem Substantiv platziert, wird das nächste Substantiv gesucht. Mit `get-noungroup-bounds` werden die Ausmasse der zugehörigen Substantivgruppe bestimmt. Diese wird anschliessend als Region[23] markiert.

`get-noungroup-bounds` verwendet prinzipiell dasselbe Vorgehen, wie wir es für *NPcat* in Abschnitt 6.4.3 gezeigt haben, ist hier allerdings in ELisp als endlicher Automat realisiert, der auf den vorher ermittelten *properties* der einzelnen Wörter operiert. Die *properties* enthalten die Wortart für jedes Wort.

Abbildung 7.7: Einbindung von NPcat *in* XEmacs*: Hervorhebung von Substantivgruppen.*



Die Markierung einer Substantivgruppe ermöglicht anschliessend verschiedene Aktionen, die prinzipiell für alle markierten Elemente möglich sind. Markierte Elemente können komplett gelöscht, kopiert oder ausgeschnitten werden. Zudem kann die Markierung verwendet werden, um die Substantivgruppe farblich hervorzuheben, wie in Abbildung 7.7 gezeigt. Da es sich hier um eine Informa-

22. Siehe http://www.xemacs.org/Documentation/21.5/html/xemacs_12.html#SEC83 (zuletzt besucht am 8.12.2010, 19:34).
23. «The text between point and the mark is known as the region.» http://www.xemacs.org/Documentation/beta/html/lispref_43.html#SEC553 (zuletzt besucht am 8.12.2010, 19:34).

tionsfunktion handelt, verschwindet diese Hervorhebung, wenn der Benutzer weiterschreibt oder eine andere Funktion aufruft.

Unter der Verwendung von *GERTWOL* (Listing 7.1 auf Seite 199 für `get-analysis`) können wir die Informationsfunktion so erweitern, dass sie den vollen Umfang von *NPcat* abdeckt. Für die hervorgehobene Substantivgruppe wird zusätzlich die Kategorie angezeigt, wie in Abbildung 7.8.

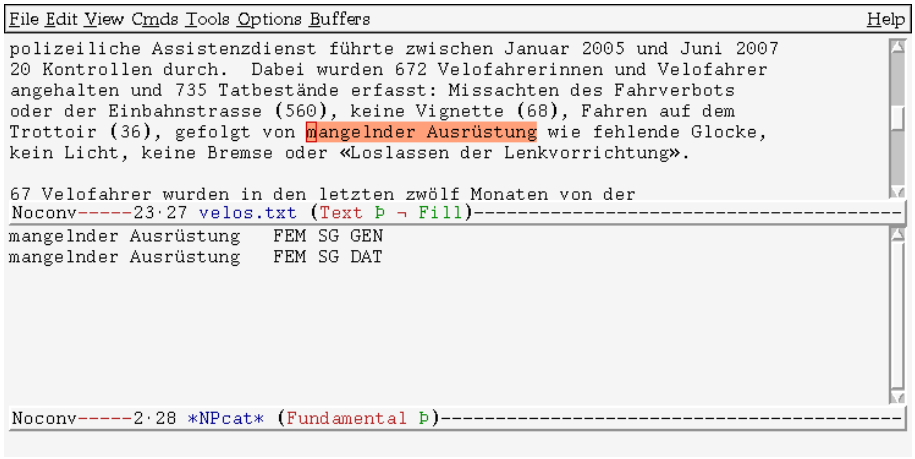


Abbildung 7.8: Einbindung von NPcat *in* XEmacs*: Hervorhebung von Substantivgruppe und Angabe ihrer Kategorie.*

Wir erlauben, dass Wortformen, deren Oberfläche sich bei der Änderung von Kasus und/oder Numerus der Substantivgruppe *nicht* ändert, die also nicht flektiert werden, in die Kennzeichnung der Wortgruppe eingeschlossen werden. Dies hat pragmatische Gründe. Die Manipulation einer Substantivgruppe erfordert das Löschen der ursprünglichen Wortgruppe an einer bestimmten Stelle im Text, das Erzeugen der gewünschten Formen der Elemente der Substantivgruppe und das Einfügen dieser Formen an der ursprünglichen Stelle im Text. Nicht veränderte Wortformen müssen erhalten bleiben. Das komplette Löschen einer Wortgruppe vom Artikel bis zum Substantiv ist einfacher und damit auch schneller – was für interaktive Prozesse ein entscheidendes Kriterium ist – als das einzelne Löschen und Ersetzen verschiedener Wortformen an verschiedenen Textstellen.

Bei der Änderung des Kasus von «*seiner kürzlich veröffentlichten Antwort*» etwa in den Akkusativ ist es also notwendig, (a) die Wortgruppe zu löschen, (b) die ursprüngliche Wortgruppe von links nach rechts zu bearbeiten, indem für Pronomen, Adjektiv und Nomen jeweils die Form für feminin, Singular, Akkusativ erzeugt wird, «*stark*» jedoch nicht geändert wird, und (c) die resultierende Folge von Wortformen an der ursprünglichen Stelle wieder einzufügen. Würde «*kürzlich*» nicht als zur Wortgruppe gehörig betrachtet, müsste man die gleichen drei Schritte zunächst für das Pronomen und dann für die Wortgruppe aus Adjektiv und Substantiv ausführen.

Am Beispiel der angeführten Manipulation – Änderung des Kasus – wird auch deutlich, dass die Tatsache, dass einige der ermittelten Substantivgruppen nicht eindeutig kategorisiert werden können, nicht per se ein Eingreifen des Benutzers erfordert. Können Genus und Numerus eindeutig ermittelt werden, ist die Substantivgruppe aber hinsichtlich Kasus ambig, ist diese Ambiguität zu vernachlässigen, wenn die Substantivgruppe in einen bestimmten (anderen) Kasus geändert werden soll. Ist die Wortgruppe hinsichtlich Kasus ambig, hinsichtlich Numerus jedoch eindeutig kategorisierbar, hängt es von den konkreten

Flexionsklassen der beteiligten Adjektive und des Substantivs ab, ob die Änderung des Numerus (von Plural in Singular oder umgekehrt) unter Verwendung aller ermittelten Kasus zu verschiedenen oder nur einer Wortformoberfläche führt. Da wir im Schreibprozess an der Oberfläche der Wortformen interessiert sind, ist es nicht notwendig, «*seiner kürzlich veröffentlichten Antwort*» hinsichtlich Kasus eindeutig zu bestimmen, wenn als Operation die Wortgruppe in den Plural gesetzt werden soll: Plural Nominativ und Plural Akkusativ sind oberflächengleich.

7.2.4 Zusammenfassung: Einbindung computerlinguistischer Ressourcen in *XEmacs*

Wir haben gezeigt, dass die in Kapitel 6 ausgewählten computerlinguistischen Ressourcen einfach in *XEmacs* eingebunden werden können. Als exemplarische Funktion zum Testen der Einbindung selbst und der Geschwindigkeit der Übergabe der zu bearbeitenden Elemente, Ausführung und Rückgabe der Ergebnisse haben wir einfache Anwendungen zur Analyse von Wortformen, Kennzeichnung von finiten Verbformen und Markierung von Substantivgruppen und Anzeige ihrer Kategorie implementiert. Diese grundlegenden Funktionen können wir nun in anderen verwenden.

7.3 Implementierung konkreter Funktionen

In diesem Abschnitt zeigen wir die konkrete Umsetzung ausgewählter Funktionen aus der Menge der in Abschnitt 4.2 beschriebenen Funktionsklassen. Für die Implementierung von Funktionen entsprechend dem Konzept des linguistisch unterstützten Redigierens sind verschiedene Kriterien zu berücksichtigen, die wir in Abschnitt 4.4 beschrieben haben.

Wir beschränken uns in der konkreten Implementierung auf Funktionen, die wenig Benutzerinteraktion erfordern und diese möglichst auf den Aufruf der Funktion beschränken, etwa auf die Angabe von Parametern. Soweit möglich, sollen die Parameter implizit – etwa über die Positionierung des Cursors – übergeben werden. Die Benennung der Funktionen ist möglichst sprechend, d. h., sie gibt wieder, was angezeigt, was geändert oder wohin sich bewegt wird. Zudem werden möglichst kurze und an den Namen der Funktion angelehnte Tastenkombinationen angegeben. Damit stehen für den Aufruf einer Funktion die für *XEmacs* üblichen Möglichkeiten zur Verfügung.

Wir stellen in den folgenden Abschnitten Beispiele für Informationsfunktionen, Bewegungsfunktionen und lokale Modifikationsfunktionen vor. Eine konkrete Funktion wird durch die Ausprägung der Eigenschaften entsprechend der Klassifizierung aus Abschnitt 4.3 definiert. Die dargestellten Funktionen wurden so ausgewählt, dass die wichtigsten Eigenschaften der repräsentierten Funktionstypen deutlich werden. Zudem sind diese Funktionen in dem Sinne prototypisch, als dass ähnliche oder verwandte Funktionen leicht aus diesen abgeleitet werden können, wie wir in Abschnitt 7.4 zeigen.

Auf globale Modifikationsfunktionen gehen wir nur kurz ein, da solche Funktionen sehr komplex sind und insbesondere erhöhte Ansprüche an involvierte computerlinguistische Ressourcen stellen, die unserer Meinung nach im Mo-

ment noch nicht erfüllt werden. Aspekte der Bedienbarkeit sprechen dafür, dass eine solche globale Modifikationsfunktion eher die kognitiven Anforderungen erhöht, als dass sie dazu beiträgt, sie zu reduzieren. Ob es also sinnvoll ist, eine solche Funktion zu verwenden, ist höchst fraglich.

Insgesamt mag die Auswahl arbiträr erscheinen. Wir wählen aus den prinzipiell vorstellbaren Funktionen jeder Klasse eine aus, aus der sich weitere ableiten lassen. Neben der Beispielhaftigkeit einer Funktion wären andere Kriterien denkbar, die sich auf empirische Daten der Schreibforschung stützen. Leider existieren unseres Wissens momentan keine solche Daten, aus denen sich ableiten liesse, welche Redigieroperationen besonders anfällig für *slips* sind – d. h., welche Redigieroperationen von einer linguistischen Unterstützung besonders profitieren würden, wie wir in Abschnitt 5.3 gezeigt haben.

Das Design von Prüf- und Korrekturprogrammen (siehe Abschnitt 3.2) basiert in der Regel auf Selbstbeobachtung und Umfragen unter einigen erfahrenen Autoren, siehe [Duffy und Robinson 1996, Sharples et al. 1989]. Gespräche – geführt bei «Computers and Composition 2009» in Davis, CA, USA, im Juni 2009 und im Online-Forum zur Konferenz sowie bei «SIG Writing 2010» in Heidelberg, Deutschland, im September 2010 – zeigen, dass erfahrene Autoren und selbst Schreiblehrer nicht über die von ihnen genutzten Werkzeuge, deren Grenzen und Möglichkeiten reflektieren, sodass wir die hier getroffene Auswahl, basierend auf Selbstbeobachtung und unter dem Aspekt der Beispielhaftigkeit, für legitim erachten.

In den folgenden Abschnitten 7.3.1 bis 7.3.5 stellen wir die Funktionen vor, die beispielhaft für eine bestimmte Klasse von Funktionen sind. Wir beschreiben sie entsprechend der Klassifizierung in Abschnitt 4.3 und stellen die Funktionsweise allgemein und anhand konkreter Redigierbeispiele dar. Für jede Funktion geben wir die allenfalls notwendigen Ressourcen an und zeigen für die Implementierung Pseudocode. Zusätzlich verdeutlichen Screenshots die Funktionsweise.

7.3.1 Beispiel für eine Informationsfunktion: Hervorhebung von Sätzen ohne finites Verb (`show-sentence-without-finverb`)

Entsprechend der Beschreibung in Abschnitt 4.2 verändern Informationsfunktionen den Text nicht, sondern verdeutlichen dem Autor bestimmte Eigenschaften des bereits geschriebenen Textes. Sehr allgemein kann man hervorhebende Informationsfunktionen mit dem *Syntaxhighlighting* für Programmiersprachen vergleichen. In Ergänzung zur Kennzeichnung bestimmter syntaktischer und morphosyntaktischer Einheiten ist es beim Verfassen natürlichsprachlicher Texte jedoch auch sinnvoll, das *Fehlen* bestimmter syntaktischer Einheiten anzuzeigen. Natürlich können Autoren bewusst auf einzelne Elemente, wie das Objekt, verzichten, um einen bestimmten stilistischen Effekt mit der so entstehenden Ellipse zu erzielen. Diese Verwendung von Sprache findet sich jedoch weitaus häufiger in fiktionalen Texten und ist dort auch eher akzeptiert als in nicht-fiktionalen Texten.

Gerade durch das Redigieren auf Satzebene, etwa das Auftrennen von sehr langen Sätzen oder das Zusammenfügen von mehreren Sätzen in einen, sind Manipulationen der Verben nötig. Unter bestimmten Umständen kann eine finite Verbform in einem Satzgefüge weggelassen werden. Werden die Satzglieder um-

gestellt oder soll das Satzgefüge in mehrere eigenständige Sätze geändert werden, muss sie jedoch explizit vorhanden sein. Während des Redigierens mag dieser Aspekt eine untergeordnete Rolle spielen, da der Fokus auf Satz(glied)ebene liegt und daher eine entsprechende Kontrolle nicht stattfindet.

In einem Redigierschritt, der explizit und bewusst erfolgt, ist es daher sinnvoll, den geschriebenen Text auf Besonderheiten hin untersuchen zu lassen. Wir beschreiben hier exemplarisch die Auszeichnung von Sätzen ohne finites Verb. So hätten Fehler wie «*Dabei es zum vielfältige Arbeitsbereiche*» (A.14 auf Seite 249) bemerkt werden können. Es ist nicht sinnvoll, diese Funktion während des Schreibens einzusetzen, da der Satz, an dem gerade geschrieben wird, natürlich ungrammatisch ist, bis er fertig geschrieben wird, und eine entsprechende Kennzeichnung dem Autor zu diesem Zeitpunkt keine zusätzliche, geschweige denn sinnvolle Information liefert.

7.3.1.1 Klassifizierung

Typ Informationsfunktion

Textveränderung nein

Spezifizität implizit durch Namen der Funktion, operiert auf Sätzen, berücksichtigt Wortart, optionales explizites Argument zum Startpunkt

Skopus Satz und Wort

Gebiet global

linguistische Ressourcen automatische Wortartenbestimmung mittels *Mbt* (dynamisch, Analyse, morphologisch)

Seiteneffekte nein

Eigener Modus nein

7.3.1.2 Funktionsweise, Beispiele

Der Autor entscheidet, wann er die Informationsfunktion aufruft, er tut dies explizit. Der Aufruf selbst enthält die notwendigen Angaben über die gewünschte Information.

Wir nennen die Funktion `show-sentence-without-finverb`. Es handelt sich um eine Informationsfunktion (`show`), die sich auf eine bestimmte linguistische Einheit – Sätze (`sentence`) – bezieht, die bestimmte Eigenschaften aufweist – das Fehlen (`without`) – von Vertretern einer bestimmten syntaktischen Kategorie – nämlich finite Verben (`finverb`).

Damit sind bereits die Einheiten benannt, die im Text ermittelt werden müssen: (a) Sätze und (b) finite Verbformen. Zu kennzeichnen sind alle Sätze, die *keine* finite Verbform enthalten, also alle «grossen» Elemente, die das «kleine» Element *nicht* enthalten. Da es sich hier um eine Informationsfunktion handelt, ist der Wechsel in einen anderen Modus nicht erforderlich – der Autor kann jederzeit Manipulationen am Text vornehmen. Die Auszeichnung kann komplett abgestellt werden, ebenso kann der Autor nach Änderungen am Text einen

erneuten Test aufrufen, ohne die vorige Auszeichnung zuvor explizit entfernen zu müssen.

Der Aufruf von `show-sentence-without-finverb` kann also jederzeit erfolgen. Das Resultat sind einige farbig hervorgehobene Sätze, wie in Abbildung 7.9. Der Autor entscheidet, ob er Änderungen an diesen Sätzen vornimmt. Bei der Navigation durch den Text (d. h. von einem dieser markierten Sätze zum nächsten) bleibt die farbige Hervorhebung erhalten.

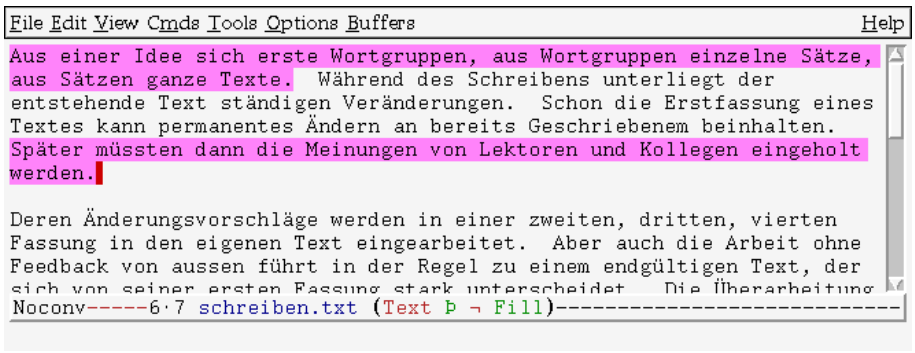


Abbildung 7.9: Beispiel für Informationsfunktion.

Der erste Satz, der in Abbildung 7.9 hervorgehoben wurde, ist tatsächlich eine korrekte Fundstelle, das finite Verb fehlt. Der zweite Satz hätte jedoch nicht hervorgehoben werden sollen. Wir sehen hier den Einfluss der Fehlerrate von *Mbt*: Im Satz «*Später müssten dann die Meinungen von Lektoren und Kollegen eingeholt werden.*» ist «*müssten*» als `VVINF`, also als Infinitiv erkannt worden, korrekt wäre die Markierung als `VMFIN`, also als finites Modalverb. Würde der Satz lauten «*Später sollten dann die Meinungen […]*», würde der Satz nicht hervorgehoben, da «*sollten*» korrekt als `VMFIN` bestimmt wird.

7.3.1.3 Implementierung

Es müssen zwei verschiedene Strukturen erkannt werden: Sätze und finite Verben. Die Funktionalität zur Erkennung von Sätzen ist in *XEmacs* bereits vorhanden (siehe Abschnitt 7.1.1.1). Anfang und Ende eines Satzes werden benötigt, um festzustellen, ob innerhalb dieser Grenzen mindestens ein finites Verb vorhanden ist, und um den Satz zu markieren – die Hervorhebung soll vom Anfangsbuchstaben des ersten Wortes bis zum Satzendzeichen reichen.

Für die Erkennung von finiten Verbformen ist es zunächst notwendig, Wortformen zur erkennen – auch diese Funktionalität ist in *XEmacs* bereits vorhanden (siehe Abschnitt 7.1.1.1). Für jede Wortform muss dann geprüft werden, ob sie eine finite Verbform ist. Es ist nicht notwendig, die exakte *Kategorie jeder* Wortform zu ermitteln, notwendig ist die Bestimmung der Wortart jeder Wortform, für Verben muss anschliessend ermittelt werden, ob es sich um ein finites oder infinites Verb handelt. Für diese Aufgabe ist der Einsatz eines Wortartenbestimmers (siehe Abschnitt 6.3) ausreichend. Wir verwenden *Mbt* (siehe Abschnitt 7.2.2) und eine abgewandelte Form der Funktion `show-finverb` (siehe Listing 7.5 auf Seite 201), die die Funktion `pos-tag-sentence` (siehe Listing 7.4 auf Seite 201) verwendet.

Beim Aufruf der Funktion kann der Autor in einem optionalen Parameter angeben, ob die Auszeichnung erst ab dem im aktuellen Textausschnitt ersten sichtbaren Wort beginnen soll. Wird kein Parameter angegeben, wird der

gesamte Text behandelt. Listing 7.8 zeigt die abstrakte Implementierung der Funktion.

Listing 7.8: Pseudocode für die Funktion `show-sentence-without-finverb`.

```
start at point (default) or
         startpoint (given by argument)
loop until end of text
  identify next sentence
  pos-tag-sentence
  foreach word
    if POS == VVFIN or VMFIN or VAFIN
      next sentence
  make extent
  highlight sentence
report "finished!"
```

7.3.1.4 Seiteneffekte, Folgefunktionen

Informationsfunktionen haben definitionsgemäss keine Seiteneffekte, da sie den Text nicht verändern. Es lassen sich abhängig von den jeweils erhaltenen Informationen Bewegungsfunktionen ableiten. Die Funktion `show-sentence-without-finverb` hebt alle Sätze ohne finites Verb hervor – der Autor entscheidet selbst, welche dieser Sätze er manipulieren möchte. Dabei muss es möglich sein, nach dem Aufruf dieser Funktion zu jedem beliebigen Punkt innerhalb des Textes zu navigieren. Beabsichtigt der Autor jedoch, tatsächlich die hervorgehobenen Sätze zu verändern und jeweils nur einen *dieser* Sätze zu bearbeiten, kann die Navigation dorthin wesentlich erleichtert werden, wenn sich der Cursor jeweils direkt zum Beginn (oder Ende) des nächsten (oder vorhergehenden) hervorgehobenen Satzes bewegen liesse.

Eine solche Bewegungsfunktion, die einer *hervorhebenden Informationsfunktion* (siehe Abschnitt 4.2) folgt, verwendet lediglich *extents* für Elemente, die bestimmten Bedingungen genügen.

7.3.2 Beispiel für eine Bewegungsfunktion: Sprung zum Beginn des nächsten finiten Verbs (`goto-next-finverb`)

Allgemein bilden Informationsfunktionen die Grundlage für komplexere Funktionen: Die Elemente, die in Informationsfunktionen hervorgehoben, gezählt oder aufgelistet werden, können als Ziele für Bewegungsfunktionen verwendet werden.

Mit der Funktion `show-sentence-without-finverb` können wir, wie oben gezeigt, Sätze hervorheben, die *keine* Wortform enthalten, die als finite Verbform kategorisiert wurde. Wir können natürlich auch explizit diejenigen Wortformen innerhalb eines Textes ermitteln, die als `VVFIN`, `VMFIN` oder `VAFIN` kategorisiert werden (wie in Listing 7.5 auf Seite 201), und diese wie in Abbildung 7.10 auf der nächsten Seite auszeichnen. Die finiten Verbformen eines Textes können zudem als Sprungziele beim Navigieren durch einen Text verwendet werden, ohne sie hervorzuheben, analog zum Bewegen zum Beginn des nächsten Satzes, wie es die bereits eingebaute Funktion M-e (`forward-sentence`) erlaubt. Die Verwendung einer solchen Bewegungsfunktion ist nicht an einen bestimmten Zeitpunkt innerhalb des Redigierprozesses gebunden, sie wird nicht in einem expliziten Modus ausgeführt.

Abbildung 7.10: Automatische Auszeichnung finiter Verbformen (Text aus der Neuen Zürcher Zeitung, 5. Februar 2010, «Portugal kämpft gegen Haushaltsdefizit», *Onlineausgabe 10:26*[24]*).*

7.3.2.1 Klassifizierung

Typ Bewegungsfunktion

Textveränderung nein

Spezifizität implizit, optionales explizites Argument für Hervorhebung des Sprungziels

Skopus Wort

Gebiet lokal

linguistische Ressourcen automatische Wortartenbestimmung mittels *Mbt* (dynamisch, Analyse, morphologisch)

Seiteneffekte nein

Eigener Modus nein

7.3.2.2 Funktionsweise, Beispiele

Der Autor verwendet Bewegungsfunktionen während des Schreibens. Mit der Bewegung des Cursors ist jeweils auch eine Veränderung des Fokus verbunden. Es ist möglich, dass zu einem anderen Punkt im gerade *sichtbaren* Textausschnitt (also im aktuellen Fenster) gesprungen wird. Ebenso ist es möglich, dass das

24. http://www.nzz.ch/nachrichten/wirtschaft/aktuell/portugal_haushaltsdefizit_1.4817202.html (zuletzt besucht am 8.12.2010, 19:34).

Sprungziel aktuell nicht sichtbar ist und sich der sichtbare Textausschnitt nach der Ausführung der Bewegung ändert.

Verwenden wir erneut den Text aus Abbildung 7.10 auf der vorherigen Seite; der Cursor befindet sich aktuell auf dem «F» von «Finanzmärkte» im oberen Drittel des Fensters. Beim (mehrmaligen) Aufruf von `goto-next-finverb` bewegt sich der Cursor nacheinander zu «verfolgen», «geht», «musste», «wird», usw. Da Bewegungsfunktionen wie Informationsfunktionen den Text nicht manipulieren und sie nicht in einem eigenen Modus ausgeführt werden, ist die Bearbeitung des Textes direkt nach dem Aufruf einer solchen Funktion möglich. Die Hervorhebung von bestimmten Elementen, also der Einbezug eines Aspektes von Informationsfunktionen, kann für bestimmte Bewegungsfunktionen nützlich sein. Denkbar ist einerseits, dass der Autor erst alle finiten Verben hervorheben lässt (so wie in Abbildung 7.10 zu sehen) und sich dann von einem hervorgehobenen finiten Verb zum nächsten bewegt. Das Verhalten wäre also ähnlich dem von `Incremental Search` (`C-s`)[25]. Es wäre optisch verdeutlicht, entlang welcher Kriterien man sich durch einen Text bewegt. Einen ähnlichen Effekt kann man erzielen, indem die Bewegungsfunktion das Ziel hervorhebt, um das Ändern des Fokus für den Autor zu erleichtern und die aktuelle Cursor-Position schneller zu finden.

In der Abbildung 7.10 findet sich im unteren Drittel eine Form von «*ankündigen*»: «*kündigte an*», hier sind Verbbasis und Verbzusatz sehr nah beieinander (vgl. Beispiel 7.16). Im oberen Drittel findet sich eine Form von «*einbrechen*»: «*brach ein*», hier steht der Verbzusatz am Satzende (vgl. Beispiel 7.17) und zudem in einer anderen Zeile als die Verbbasis.

(7.16) Er **kündigte an**, den Vorstoss mit allen rechtlichen und politischen Mitteln stoppen zu wollen.

(7.17) Die Börse in Lissabon **brach** daraufhin am Donnerstag um fast fünf Prozent **ein**.

Für Verbgefüge wird der Cursor auf den Anfang der Verbbasis platziert. Gerade für diesen Fall ist eine Hervorhebung sinnvoll, die den Verbzusatz einschliesst. Denkbar ist ein Szenario, in dem der Autor von Verb zu Verb durch den Text navigiert, um zu entscheiden, ob eines geändert werden sollte (abhängig von jeweils vorher verwendeten Verben, einem allgemeinen stilistischen Bedürfnis, o. ä.) – hier ist das Risiko hoch, dass ein sehr weit entfernt von der Verbbasis platzierter Verbzusatz ausserhalb des Fokus ist und darum beim Ändern vergessen wird, wie vermutlich geschehen im Beispiel «*Der Verband unabhängiger Schweizer Hochzeitsplaner bietet einen Diplomlehrgang zum Hochzeitsplaner durch.*» (siehe A.203 auf Seite 297).

Wird das optionale Argument zur Hervorhebung des Verbs (für Verbgefüge schliesst dies auch die Markierung und Hervorhebung des Verbzusatzes ein) gesetzt, wird nach Erreichen des Navigationsziels das Verb (und der Verbzusatz) hervorgehoben. Wird die Funktion erneut aufgerufen, erlischt die Hervorhebung an der bisherigen Cursor-Position (ebenso die Hervorhebung eines allfälligen Verbzusatzes) und das neue Navigationsziel wird hervorgehoben.

25. Siehe http://www.xemacs.org/Documentation/21.5/html/xemacs_15.html#SEC116 (zuletzt besucht am 8.12.2010, 19:34).

Wird nach dem Aufruf der Funktion eine beliebige andere Funktion aufgerufen oder Text eingegeben oder gelöscht, erlischt die Hervorhebung ebenfalls.

7.3.2.3 Implementierung

Für die Implementierung sind nur Informationen über Wortformen notwendig, grössere Strukturen müssen nicht erkannt werden. Von jeder Wortform, die nach der aktuellen Cursor-Position vorhanden ist, wird ermittelt, ob es sich um ein finites Verb handelt und somit als Sprungziel in Frage kommt. Als Ressource wird darum lediglich *Mbt* benötigt.

Listing 7.9 zeigt den Pseudocode für die Funktion. Wie schon für die Informationsfunktion `show-sentence-without-finverb` wird das Ergebnis der Funktion `pos-tag-sentence` verwendet. Im Gegensatz zur Informationsfunktion, die global arbeitet, wird die Bewegungsfunktion auf kleinerem Raum eingesetzt, es müssen nur Wortformen behandelt werden, die im Text *nach* der aktuellen Cursor-Position auftreten. Ist ein finites Verb gefunden, ist die Funktion beendet. Es wird keine explizite Meldung über den Abschluss der Funktion erwartet, das Ende wird implizit dadurch markiert, dass der Cursor an einer neuen Position sichtbar wird. Die Angabe, ob das Sprungziel hervorgehoben werden soll, ist ein optionaler Parameter.

Listing 7.9: Pseudocode für die Funktion `goto-next-finverb`.

```
start at point
include highlighting?
loop until "finished"
    pos-tag-sentence
    foreach word
       if POS == VVFIN or VMFIN or VAFIN
          set point to word
          (make extent
           highlight extent
           process next words
             if POS == PTKVZ
             make extent
             highlight extent)
          set "finished"
       else
          next word
    next sentence
```

Je nach ursprünglicher Position des Cursors kann sich dabei gleichzeitig der sichtbare Textausschnitt verändern. Steht der Cursor beispielsweise in der letzten Zeile des Textes aus Abbildung 7.10 auf Seite 210 («*Manuela Ferreira Leite, umzustimmen.*»), ist das nächste finite Verb nicht sichtbar, d. h., nach Abschluss der Funktion verändert sich der Textausschnitt, sodass der Cursor auf einem im aktuellen Fenster sichtbaren Wort platziert ist. Wie bereits erwähnt, kann die Veränderung des Fokus erleichtert werden, indem das finite Verb (inklusive eines etwaigen Verbzusatzes) hervorgehoben wird.

7.3.2.4 Seiteneffekte, Folgefunktionen

Bewegungsfunktionen haben keine Seiteneffekte. Die optionale Hervorhebung des Navigationsziels erlischt beim nächsten Tastendruck oder Aufruf einer anderen Funktion. Im Beispiel, wie in Abbildung 7.10 auf Seite 210 gezeigt, sehen wir auch ein Beispiel für Fehler der von uns verwendeten Ressource *Mbt*: Im Satz «*Es gibt im Leben Grenzen, die wir nicht überschreiten können.*» (im letzten

sichtbaren Absatz) ist lediglich «*gibt*» als finite Verbform ausgezeichnet, jedoch nicht «*können*». Dies liegt daran, dass Mbt «*können*» als infinite Verbform kategorisiert. Computerlinguistische Ressourcen arbeiten, wie in Kapitel 6 ausführlich besprochen, nicht absolut fehlerfrei, wie wir es für eine optimale und zuverlässige Unterstützung benötigen würden. Im Fall von Informations- und Bewegungsfunktionen mag dieser Fakt jedoch akzeptabel sein, da diese Funktionen den Text nicht manipulieren und somit keine Fehler in den Text eingefügt werden.

7.3.3 Beispiel für eine Modifikationsfunktion: Vertauschen von Konjunkten (`transpose-conjuncts`)

Das Vertauschen der Elemente einer Koordination ist eine triviale Operation. Die Redigieranweisung für einen Autor ist sehr simpel: Die Konjunkte sollen vertauscht werden. Für Autoren ist es relativ einfach, die beteiligten Konjunkte zu identifizieren, auch wenn diese aus mehr als einer Wortform bestehen und nicht symmetrisch sind wie in «***Der effektive Nutzen*** *und* ***die Risiken*** *bleiben umstritten*.» Das linke Konjunkt besteht aus einer Substantivgruppe mit drei Wortformen, das rechte Konjunkt ist eine Substantivgruppe mit zwei Wortformen. Koordinationen wie «*erfordert jedoch* ***hohe Konzentration*** *und* ***sehr viele auszuführende Funktionen***» sind ebenfalls nicht symmetrisch. Der Autor kann die einzelnen Bestandteile einer Aufzählung (oder einer Alternative) – linkes Konjunkt, Konjunktion, rechtes Konjunkt – mittels Maus oder Tastatur eindeutig markieren. Das Vertauschen des linken und rechten Konjunkts ist jedoch nicht trivial, wie bereits in Abschnitt 7.1.2 gezeigt.

Wenn wir die einzelnen Elemente einer Koordination so markieren können, dass linkes Konjunkt, Konjunktion und rechtes Konjunkt eindeutig bestimmt sind, kann auf diese Informationen gezielt zugegriffen werden und linkes und rechtes Konjunkt können vertauscht werden. Mittels Maus oder Tastatur kann jeweils nur eine *zusammenhängende* Folge von Zeichen markiert weden, wir müssen jedoch drei Zeichenfolgen explizit markieren.

Am wenigsten Bedienerinteraktion würde es erfordern, wenn linkes und rechtes Konjunkt automatisch bestimmt werden könnten. Allerdings sind koordinierende Strukturen «[…] a pernicious source of structural ambiguity» [Resnik 1999, S. 96] und deren automatische Analyse noch lange keine gelöste Aufgabe: «Conjunctions are about the hardest things in parsing, and we have no grip on exactly what it takes to help parse them.» [McClosky et al. 2006, S. 157] Aktuelle Forschung auf dem Gebiet der Koordinationsbestimmung zeigt, dass die Qualität der automatischen Bestimmung von Konjunkten nicht zufriedenstellend ist. «Despite the frequency and implied importance of coordinative structures, coordination disambiguation remains a difficult problem even for state-of-the-art parsers.» [Hara et al. 2009, S. 967] Selbst die hypothetische optimale Kombination verschiedener Systeme (*oracle*) erreicht lediglich eine Genauigkeit von ca. 64 %, wie Ogren [2010] zeigt. Dies ist nicht ausreichend, es ist Interaktion mit dem Benutzer notwendig. Wir gehen daher davon aus, dass die Grösse der Konjunkte vom Autor bestimmt wird und bieten ihm dafür Hilfestellungen.

Das Vertauschen von Konjunkten erfordert keine linguistischen Ressourcen. Wir nutzen die Tatsache aus, dass eine Koordination aus linkem Konjunkt, Konjunktion und rechtem Konjunkt besteht und Wortformen voneinander

durch Leerzeichen oder Interpunktion getrennt werden. Die Konjunktion ist eine einzelne Wortform, linkes und rechtes Konjunkt können einzelne Wortformen oder Wortgruppen sein, sie müssen nicht symmetrisch (d. h. von der gleichen Wortart und gleich lang) sein. Diese Tatsache trifft sogar für weitere Sprachen ausser Deutsch zu. Wir haben hier also eine relativ sprachunabhängige Funktion.

Mit dieser Funktion berücksichtigen wir auch den kreativen Aspekt des Schreibprozesses, das «Spielen» mit Formulierungen. Das Tauschen von Konjunkten soll in einem eigenen Modus erfolgen, der nur die Auswahl der Konjunkte und das Vertauschen ermöglicht. Beide Operationen können beliebig oft und in beliebiger Reihenfolge ausgeführt werden. Es ist jedoch nicht möglich, andere Funktionen aufzurufen oder Text einzugeben oder zu löschen. Das Austesten verschiedener Varianten einer Koordination erachten wir als so komplex, dass wir die Unterbrechung dieses Vorgangs verhindern wollen, indem wir einen gesonderten Modus verwenden. Natürlich kann der Modus jederzeit verlassen werden, ohne eine Änderung vorgenommen zu haben.

7.3.3.1 Klassifizierung

Typ Modifikationsfunktion

Textveränderung ja

Spezifizität implizit über Namen der Funktion, explizite interaktive Bestimmung der Grösse der Konjunkte

Skopus Wörter und Wortgruppen

Gebiet lokal

linguistische Ressourcen keine

Seiteneffekte möglich

Eigener Modus ja

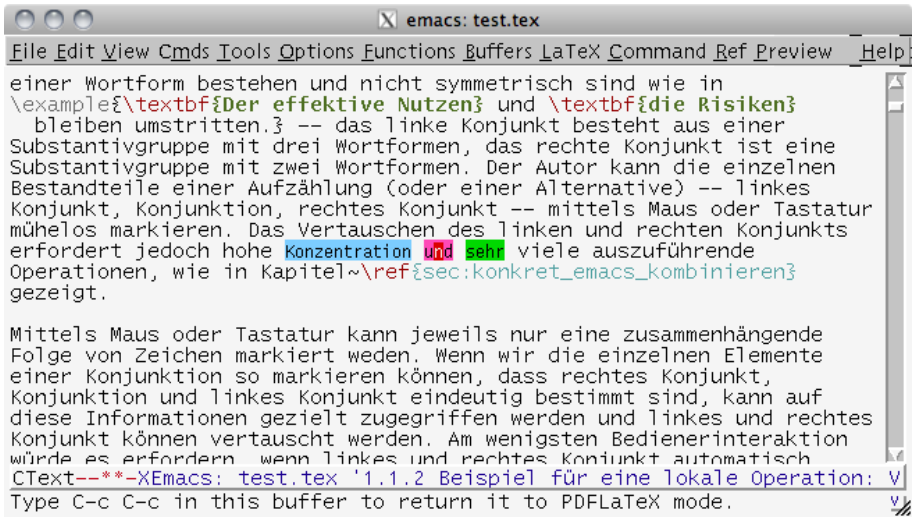
7.3.3.2 Funktionsweise, Beispiele

Platziert der Autor den Cursor auf einer Konjunktion und ruft dann den `conjunct-mode` auf, erscheint das aktuelle Textfenster wie in Abbildung 7.11 auf der nächsten Seite. Eingeben und Löschen von Text sowie der Aufruf anderer Funktionen ist in diesem Modus deaktiviert, es ist nur die Manipulation der Koordination möglich. Beim Start des Modus sind die Konjunktion (pink hinterlegt) sowie die linke anschliessende Wortform (blau hinterlegt) und die rechte anschliessende Wortform (grün hinterlegt) ausgezeichnet.[26]

Handelt es sich um einstellige Konjunkte wie in «*lesen und schreiben*» entspricht die blaue und grüne Auszeichnung tatsächlich den Konjunkten, sonst sind nur Teilelemente ausgezeichnet. Für Konstruktionen wie in unserem Beispiel sind die Konjunkte grösser als jeweils nur eine Wortform. Der Autor kann sowohl die

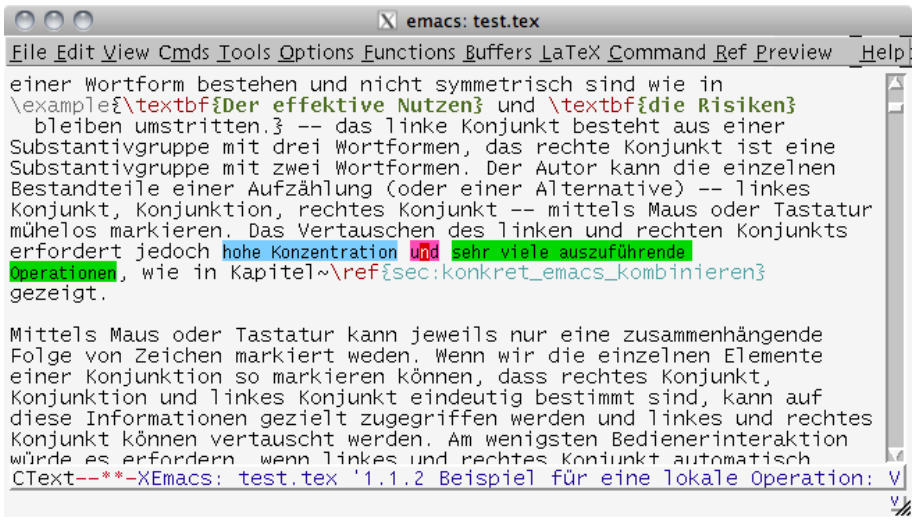
26. Die gewählten Farben für die Hinterlegung sind wie auch für die Informationsfunktion in Abschnitt 7.3.1 provisorisch. Die Farbe lässt sich einerseits eigenen Bedürfnissen und Gegebenheiten (Rot-Grün-Blindheit) anpassen und sollte andererseits akzeptierten Grundsätzen (rot steht für etwas Negatives etc.) folgen.

Abbildung 7.11: Aufruf des `conjunct-mode`.



linke als auch die rechte Markierung nach links bzw. nach rechts wortformweise erweitern und so die vollständigen Konjunkte markieren, siehe Abbildung 7.12.

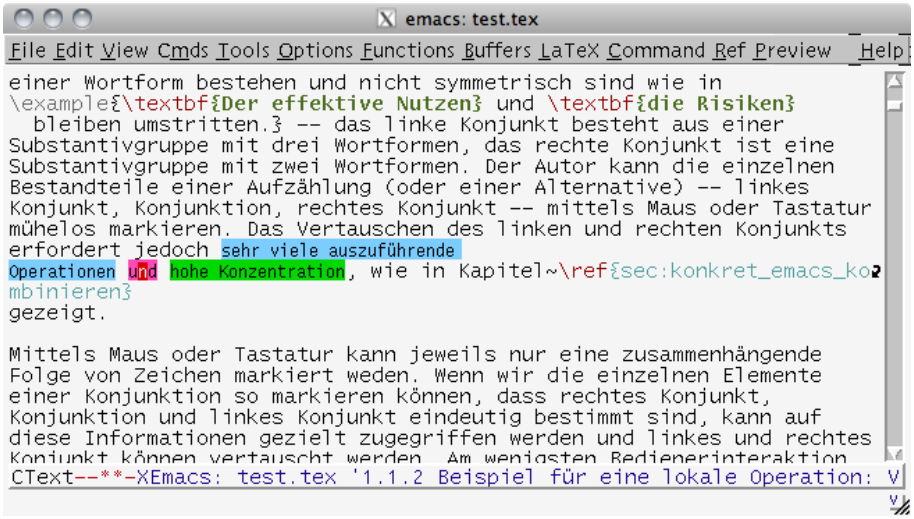
Abbildung 7.12: Markieren des linken und rechten Konjunkts, direkt an die Konjunktion anschliessend.



Hierzu sind explizite Tastenkombinationen vorhanden. Wurde ein Konjunkt «zu weit» gewählt, kann diese Auswahl ebenfalls über eine einfache Tastenkombination rückgängig gemacht werden. So erweitert das Drücken der Taste l das linke Konjunkt um eine Wortform nach links, die Tastenkombination Shift-l verkleinert das linke Konjunkt von links um eine Wortform. Es ist nicht möglich, die Markierung eines Konjunkts vollständig zu löschen oder ein Konjunkt über die Satzgrenze hinaus zu erweitern. Wie in Abbildung 7.12 zu sehen, spielen Zeilenumbrüche keine Rolle, das rechte Konjunkt setzt sich in der nächsten Zeile fort.

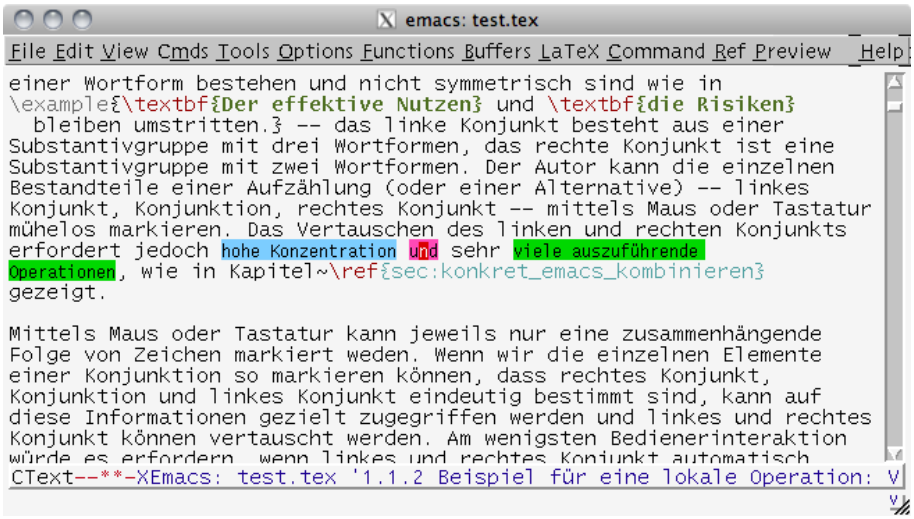
Anschliessend können beide Konjunkte durch Drücken der Taste t getauscht werden. Dabei erhält das neue linke Konjunkt die blaue Hinterlegung, das neue rechte Konjunkt ist grün hinterlegt, siehe Abbildung 7.13 auf der nächsten Seite. Mit dem Tauschen der Konjunkte ist der Modus nicht automatisch beendet. Der Autor kann beide Konjunkte durch erneutes Drücken der Taste t erneut tauschen oder beide Konjunkte weiter vergrössern oder verkleinern und anschliessend erneut tauschen etc.

Abbildung 7.13: Vertauschen des linken und rechten Konjunkts, direkt an die Konjunktion anschliessend.



In unserem Beispiel «*hohe Konzentration und sehr viele auszuführende Operationen*» ist auch denkbar, dass die beiden Konjunkte «*hohe Konzentration*» und «*viele auszuführende Operationen*» sind, dass also «*sehr*» nicht im rechten Konjunkt enthalten ist. Diese Markierung der Konjunkte ist ebenfalls möglich, wie in Abbildung 7.14 gezeigt. Dazu wird hier das rechte Konjunkt von links verkleinert.

Abbildung 7.14: Markieren des linken und rechten Konjunkts, nicht direkt an die Konjunktion anschliessend.



Es ist möglich, diese «Reduktion» wieder rückgängig zu machen, d. h., das rechte Konjunkt wortformweise nach links zu erweitern. Es ist nicht möglich, das rechte Konjunkt über die Konjunktion hinaus nach links zu erweitern oder die Markierung komplett zu löschen. Gleiches gilt analog für das linke Konjunkt. Der Tausch dieser Konjunkte führt zu einem Resultat wie in Abbildung 7.15 auf der nächsten Seite. Es werden tatsächlich nur die markierten Konjunkte getauscht, Konjunktion und nicht markierte Wortformen ändern ihre Position im Text nicht. So wäre es auch möglich, lediglich die Substantive von zwei Substantivgruppen oder nur die Adjektive zweier Wortgruppen zu tauschen.

Nach dem Verlassen des Modus stehen wieder alle gewohnten Funktionen zur Verfügung und die Hervorhebungen sind entfernt.

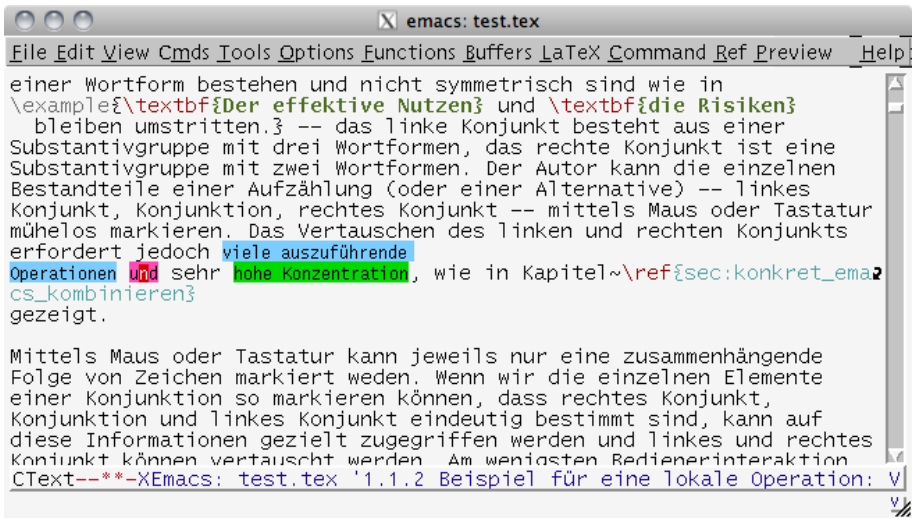


Abbildung 7.15: Vertauschen des linken und rechten Konjunkts, nicht direkt an die Konjunktion anschliessend.

7.3.3.3 Implementierung

Für die Implementierung werden keine externen Ressourcen benötigt, wir verwenden lediglich die Erkennung von Wortformen und Satzgrenzen. Beim Aufruf des `conjunct-mode` wird die Wortform, auf der der Cursor positioniert ist, als *Konjunktion* betrachtet. Die Wortform links davon ist das *linke Konjunkt*, die Wortform rechts vom Cursor ist das *rechte Konjunkt*. Jeder Wortform wird ein *extent* zugeordnet und sie werden entsprechend farblich hinterlegt.

Listing 7.10: Pseudocode für die Funktionen des `conjunct-mode`.

```
switch to conjunct-mode
start at point
make extent for conjunction, left conjunct, right conjunct
highlight extents
allow several executions (in mixed order) of
  modify extent of left conjunct
       (be aware of sentence-start and conjunction)
  modify extent of right conjunct
       (be aware of sentence-end and conjunction)
  transpose conjuncts
quit conjunct-mode
```

Listing 7.10 zeigt die abstrakte Implementierung der Funktion. Wir definieren verschiedene Funktionen zur wortformweisen Erweiterung und Verkleinerung der *extents* für die Konjunkte:

- Das linke Konjunkt kann nach links erweitert (l) und von links verkleinert (Shift-l) werden.
- Das rechte Konjunkt kann nach rechts erweitert (r) und von rechts verkleinert (Shift-r) werden.
- Wenn das linke Konjunkt um mindestens eine Wortform nach links vergrössert wurde, kann es von rechts verkleinert (C-l) und nach rechts vergrössert (M-l) werden.
- Wenn das rechte Konjunkt um mindestens eine Wortform nach rechts vergrössert wurde, kann es von links verkleinert (C-r) und nach links vergrössert (M-r) werden.

Vergrösserungen und Verkleinerungen können nicht über die Konjunktion sowie über Satzanfang (für das linke Konjunkt) und Satzende (für das rechte Konjunkt) hinaus erfolgen. Die Markierung eines Konjunkts kann nicht vollständig entfernt werden, d. h., ein *extent* kann nicht gelöscht werden. Zu jedem beliebigen Zeitpunkt kann im `conjunct-mode` das Vertauschen der aktuell bestimmten Konjunkte mittels t aufgerufen werden.

Wie in *XEmacs* üblich, wird der `conjunct-mode` mit der Tastenkombination C-c C-c verlassen. Die Hervorhebungen werden entfernt, der Text ist in der zuletzt gewählten Anordnung sichtbar und alle Funktionen stehen wieder zur Verfügung.

7.3.3.4 Seiteneffekte, Folgefunktionen

Modifikationsfunktionen verändern den Text. Die Korrektheit der bearbeiteten Koordination selbst wird im `conjunct-mode` durch den Autor direkt gesteuert. In unserem Beispiel haben wir eine Koordination bearbeitet, bei der das Vertauschen der Konjunkte keinen Einfluss auf die syntaktischen Beziehungen zu anderen Wortgruppen hatte. Es ist jedoch möglich, die zu tauschenden Konjunkte so auszuwählen, dass der ehemals wohlgeformte Satz nicht mehr wohlgeformt ist. Wird eine Koordination in einem noch nicht vollständigen Satz umsortiert, sind syntaktische Beziehungen zu anderen Satzgliedern zu vernachlässigen oder bestehen gar nicht.

Im Beispielsatz gibt es noch eine weitere Koordination: «*Das Vertauschen des linken und rechten Konjunkts erfordert jedoch hohe Konzentration und sehr viele auszuführende Operationen …*». Je nach Sichtweise kann die erste Koordination dieses Satzes lauten «*des linken und rechten Konjunkts*», «*linken und rechten Konjunkts*» oder «*linken und rechten*». Nur beim Tauschen der Konjunkte für den letzten Fall ist keine anschliessende Bearbeitung des Satzes nötig, in allen anderen Fällen muss nach dem Tauschen der Konjunkte noch weiter korrigiert werden. Zudem finden wir hier eine Bestätigung unserer Ausgangsannahme, dass die Konjunkte einer Koordination nicht automatisch bestimmt werden können. Und wir finden ebenfalls eine Instantiierung des Zitats vom Beginn dieses Kapitels: «This is the proper use of an imperfect rule-driven program: its output is expected to be examined by a person.» [Hamlet 1986]

Die Nachbearbeitung der «Umgebung» der Koordination hängt davon ab, wie diese «Umgebung» gestaltet ist: Ist der Satz selbst noch nicht vollständig geschrieben – weil der Autor während des Schreibens eine Änderung der gerade geschriebenen Koordination vornimmt –, ist eine Korrektur möglicherweise gar nicht notwendig, weil betroffene Elemente des mental bereits vorhandenen vollständigen Satzes noch nicht manifestiert sind und die Korrektur damit mental erfolgt. Das tatsächliche Ausmass notwendiger anschliessender Anpassungen kann also nicht im Voraus bestimmt werden. Es hängt von der Schreibstrategie des Autors, der Auswahl der beteiligten Konjunkte und der Häufigkeit der Verwendung der Funktion ab.

7.3.4 Beispiel für eine Modifikationsfunktion: Ändern des Numerus einer Substantivgruppe (`toggle-noungroup-number`)

Wir zeigen eine Modifikationsfunktion, die computerlinguistische Ressourcen verwendet. In Abschnitt 7.2.3 haben wir die Einbindung unserer Ressource *NPcat* zur Extraktion von Substantivgruppen und Bestimmung ihrer morphosyntaktischen Eigenschaften – ihrer Kategorie – gezeigt. In Abschnitt 6.4 haben wir begründet, warum es notwendig ist, für unsere Zwecke eine eigene Ressource für diese Aufgaben zu entwickeln. Wir verwenden für *NPcat* die Ressourcen *Mbt* und *GERTWOL/GERGEN*. Als Beispiel für eine lokale Modifikationsfunktion unter Verwendung von *NPcat* wählen wir die Änderung des Numerus von Substantivgruppen.

Die Änderung des Numerus für eine Substantivgruppe mag verschiedene Gründe haben, über die wir hier nicht spekulieren wollen. Interessant ist die Pluralisierung oder Singularisierung von Substantivgruppen mit mehreren Elementen – die Änderung des Numerus eines einzelnen Substantivs ist durch den Autor direkt einfacher und schneller zu realisieren. Eine Substantivgruppe wird durch das Substantiv bestimmt, d. h., der Fokus des Autors liegt vermutlich auf dem Substantiv. Soll eine morphosyntaktische Eigenschaft einer Substantivgruppe verändert werden, besteht daher die Gefahr, dass nach der Änderung des Substantivs die Änderung der gesamten Substantivgruppe als abgeschlossen betrachtet wird.

Wir verwenden einen `electric-buffer-mode`: Durch das Aufrufen der Funktion wird ein Modus aktiviert, der – im Gegensatz etwa zum `conjunct-mode` – nicht explizit beendet werden muss. Führt der Benutzer eine Aktion aus, die nicht zu der Funktion gehört, wird der Modus ohne jegliche Änderung des Textes abgebrochen. Wählt der Benutzer eine vorgeschlagene Variante zur Änderung aus, wird diese Änderung umgesetzt und der Modus anschliessend automatisch beendet.

Es handelt sich um eine lokale Modifikationsfunktion, da sie lediglich auf eine Veränderung innerhalb einer Substantivgruppe beschränkt ist. Der Autor bezeichnet den Ort der Operation implizit durch die Positionierung des Cursors – damit sind der Satz, in dem die Substantivgruppe sich befindet, und die Substantivgruppe selbst verortet. Die Ausdehnung der Substantivgruppe von der Cursor-Position aus nach links – d. h. der Beginn der Substantivgruppe – und nach rechts – d. h. das Ende (also das Substantiv) der Substantivgruppe – wird automatisch bestimmt. Die Ausdehnung – d. h. die Elemente der Wortgruppe – wird hervorgehoben, sodass jederzeit visualisiert wird, was verändert wird. Wir haben dies schon für die Funktion `mark-noungroup` in Abschnitt 7.2.3 gezeigt. Die Modifikationsfunktion sorgt dafür, dass wirklich alle Elemente der Substantivgruppe konsistent verändert werden, sodass die Kongruenz innerhalb der Wortgruppe gewahrt ist. Es werden also Fehler verhindert, bei denen nur das Substantiv oder nur ein Adjektiv einer Substantivgruppe im Numerus verändert werden und die übrigen Elemente ihre ursprüngliche Form behalten.

7.3.4.1 Klassifizierung

Typ Modifikationsfunktion

Textveränderung ja

Spezifizität implizit über Funktionsaufruf, explizite interaktive Bestimmung der Grösse der Nominalphrase

Skopus Wort, Wortgruppe

Gebiet lokal (Satz)

linguistische Ressourcen Bestimmung der Bestandteile von Substantivgruppen und deren Kategorie mittels *NPcat* (Wortartenbestimmung mittels *Mbt* und morphologische Analyse mittels *GERTWOL*), Generierung mittels *GERGEN* (dynamisch, Analyse und Generierung, morphologisch und syntaktisch)

Seiteneffekte möglich

Eigener Modus electric mode

7.3.4.2 Funktionsweise, Beispiele

Der Autor platziert den Cursor innerhalb der Substantivgruppe, die verändert werden soll, und ruft die Funktion `toggle-noungroup-number` auf. Die Funktion verhält sich ähnlich der Funktion `toggle-number`, siehe Listing 7.2 auf Seite 199. Der *electric* Modus wird gestartet und es öffnet sich ein neuer Buffer. Der Cursor wird automatisch dorthin platziert und ist nicht mehr innerhalb der Substantivgruppe platziert, die jedoch farblich hervorgehoben ist (wie in Abbildung 7.7 auf Seite 203 schon gezeigt). Stellt der Benutzer fest, dass er vor Beenden der Funktion eine andere Änderung im Text vornehmen möchte und bewegt den Cursor aus diesem *electric* Buffer zurück in den Buffer mit dem Text, erlischt der *electric* Buffer und die Funktion wird ohne weitere Auswirkungen abgebrochen.

Mit dem Starten der Funktion wird die Substantivgruppe markiert und hervorgehoben, im *electric* Buffer werden die Lesarten der Substantivgruppe aufgelistet, wie schon in Abbildung 7.8 auf Seite 204 gezeigt. Für jede Lesart wird die entsprechende Abwandlung hinsichtlich Numerus vorgenommen und ebenfalls angezeigt, siehe Abbildung 7.16.

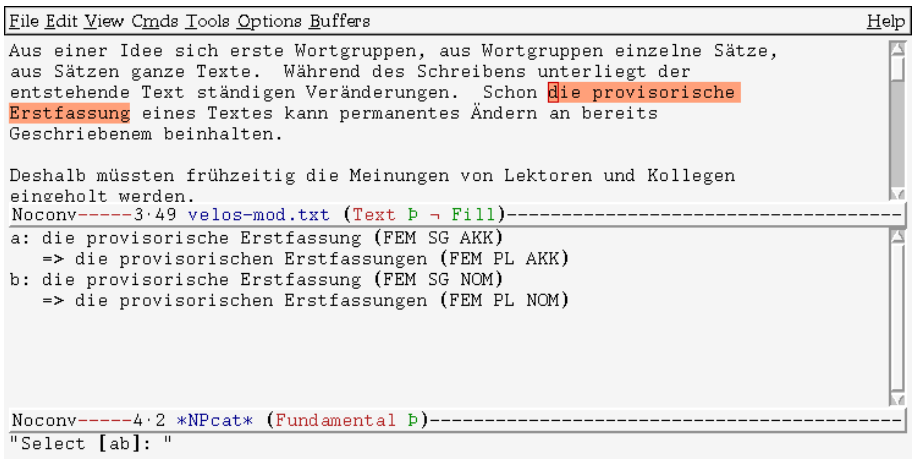


Abbildung 7.16: Auswahl der gewünschten Form für `toggle-noungroup-number`.

Für Substantivgruppen, die hinsichtlich Numerus ambig sind, etwa `MASC SG NOM, MASC PL GEN` («*abgelehnter Asylbewerber*» oder «*dieser Behälter*»), erhalten wir mehrere Varianten der Substantivgruppe, nämlich `MASC SG NOM`

und `MASC SG GEN` («*abgelehnte Asylbewerber*» und «*abgelehnten Asylbewerbers*» bzw. «*diese Behälter*» und «*dieses Behälters*»). Der Benutzer platziert den Cursor auf die Zeile mit der korrekten Analyse und der gewünschten Variante der Substantivgruppe und wählt diese durch Drücken von SPC aus. Ist die Kategorie der Substantivgruppe eindeutig bestimmbar, enthält der *electric* Buffer nur eine Zeile. Die Änderung wird automatisch vorgenommen und der *electric* Buffer erlischt; die Funktion ist beendet. Die Funktion kann auch explizit durch Drücken von q abgebrochen werden.

7.3.4.3 Implementierung

Listing 7.11 zeigt die abstrakte Implementierung der Funktion. Wir verwenden `get-noungroup` (siehe Listing 7.7 auf Seite 203). Da wir für *NPcat* sowohl *Mbt* als auch *GERTWOL/GERGEN* benötigen und beide eventuell erst gestartet werden müssen, ist mit einer kurzen Wartezeit zu rechnen. Wurden beide Prozesse bereits zuvor für andere Funktionen benötigt und laufen daher im Hintergrund, entsteht keine Verzögerung.

Listing 7.11: Pseudocode für die Funktion `toggle-noungroup-number`.

```
switch to electric-number-mode
get-noungroup
toggle number
allow choosing noungroup
replace noungroup at point with selected noungroup
quit electric-number-mode
```

7.3.4.4 Seiteneffekte, Folgefunktionen

Diese Funktion kann Seiteneffekte haben. Stand die Substantivgruppe ursprünglich im Nominativ Singular, handelt es sich vermutlich um das Subjekt. Durch die Pluralisierung geht die Kongruenz mit der finiten Verbform – sofern diese schon geschrieben ist – verloren und muss wieder hergestellt werden. Zudem können – wie für andere Kasus auch – (Relativ)pronomen auf diese Substantivgruppe verweisen, dann muss auch hier die Kongruenz geprüft und gegebenenfalls korrigiert werden. Sollten diese Korrekturen automatisch erfolgen, müsste der Satz tatsächlich schon vollständig geschrieben sein, zudem würden wir tiefe syntaktische Analyse benötigen – was wir in Abschnitt 4.2 bereits als unmöglich eingeschätzt haben. Diese Anpassungen müssen also vom Autor selbst vorgenommen werden – wofür auch wieder entsprechende Modifikationsfunktionen (etwa Pluralisieren einer Verbform) verwendet werden können. Zur automatischen Bestimmung der zugehörigen Pronomen müssten wir Anaphernresolution verwenden, was heute nicht besonders zuverlässig funktioniert [Klenner et al. 2010a] und sich daher ebenfalls verbietet.

Diese Seiteneffekte treten natürlich auch auf, hätte der Autor die fragliche Substantivgruppe selbst – also ohne die Verwendung linguistisch basierter Funktionen – verändert. Es handelt sich also nicht um durch die Funktion verursachte Seiteneffekte, sondern um durch das Redigieren verursachte Seiteneffekte.

Die Funktion kann jederzeit aufgerufen werden: während des Schreibens oder in einer expliziten Redigierphase. Wir können nicht voraussehen, welche Operationen der Autor anschliessend an die Pluralisierung plant: Wird die gesamte Substantivgruppe zusätzlich im Kasus geändert? Wird das Substantiv gegen ein

anderes getauscht? Wird die Wortgruppe an einen anderen Ort, evtl. in einen anderen Satz verschoben? Ist die Substantivgruppe Teil einer Koordination und wird mit einem anderen Konjunkt getauscht? Davon abhängig können Seiteneffekte, die direkt nach dem Ändern des Numerus aufgetreten sind, wieder hinfällig werden – eine automatische Korrektur würde also den Arbeitsprozess des Autors unterbrechen und behindern.

Wir finden hier eine Verdeutlichung des Prinzips, dass das Textbearbeitungsprogramm und dessen Funktionen lediglich Werkzeuge sind, die der Autor nach seinen Bedürfnissen verwendet, entsprechend dem ersten Zitat zu Beginn dieses Kapitels: «It is very important that the user feel in control of the editor and not the other way around.»Steven R. Wood [Wood 1981]. Und das Ziel dieser Funktion ist die Änderung des Numerus für eine Substantivgruppe – der Autor erwartet und wünscht nicht, dass das System selbständig noch andere nicht intendierte Änderungen am Text durchführt, sondern ihn bei der *gewünschten* Änderung optimal unterstützt.

7.3.5 Beispiel für eine Modifikationsfunktion: Ersetzen eines Verbs (`replace-verb`)

Neben dem *Verändern* morphosyntaktischer Eigenschaften linguistischer Einheiten, wie für die Funktion `toggle-noungroup-number` gezeigt, ist auch der *Austausch* von Wortgruppen oder Wörtern eine Redigieroperation. Wir zeigen das lokale Ersetzen eines Verbs durch ein anderes – die morphosyntaktischen Eigenschaften des ursprünglichen Verbes sollen dabei erhalten bleiben. Es ist also zunächst notwendig, diese zu ermitteln und dann die entsprechende Wortform des Ersatzverbs zu generieren. Anschliessend wird die ursprüngliche Verbform durch die neue Verbform ersetzt. Dabei können verschiedene Fälle eintreten.

Gehen wir zunächst davon aus, dass das ursprüngliche Verb ein Simplex, eine Derivation oder ein Kompositum ist, jedoch kein Verbgefüge – d. h., wir müssen keine Getrenntschreibung von Verb und Verbzusatz in einigen syntaktischen Konstellationen berücksichtigen. Besitzt die ursprüngliche Verbform eine Kategorie aus Präsens oder Präteritum Indikativ oder Konjunktiv, handelt es sich um eine einzelne Wortform. Geht es jedoch um Formen des Perfekts, des Plusquamperfekts, des Futurs I oder II (Indikativ oder Konjunktiv), sind mehrere Wortformen involviert. Die entsprechenden Formen bestehen jeweils aus einer Verbform von «*haben*», «*sein*» oder «*werden*» im Präsens oder Präteritum und dem Partizip II des Verbs oder dem Infinitiv. Alle Verben bilden das Futur II mit dem Hilfsverb «*werden*», dem Partizip II des Verbs und «*haben*» oder «*sein*». Verben unterscheiden sich hinsichtlich der Verwendung von «*sein*» oder «*haben*» für die Bildung von Perfekt- und Plusquamperfektformen und Futur II. Stimmen sowohl das ursprüngliche als auch das Ersatzverb hinsichtlich dieser Eigenschaft überein, betrifft die Ersetzung lediglich die Partizipform. Bildet eines der Verben die Perfekt- und Plusquamperfektformen jedoch mit «*haben*» und das andere mit «*sein*», muss zusätzlich das Hilfsverb ersetzt werden.

Für Modalkonstruktionen wie «*Spuren davon **könnten** die Leukämie-Fälle bei Kosovo-Soldaten **erklären**.*», in dem «*erklären*» durch «*beweisen*» ersetzt werden soll, muss das Modalverb nicht verändert werden. Es wird lediglich der Infinitiv von «*erklären*» durch den Infinitiv von «*beweisen*» ersetzt. Gleiches gilt für Formen wie «*Somit **könne** eine Verunreinigung mit gefährlichem Plutonium nicht mehr **ausgeschlossen werden**.*», wenn «*ausschliessen*» durch «*vermeiden*» ersetzt

werden soll. Hier muss ebenfalls lediglich «*ausgeschlossen*» durch «*vermieden*» ersetzt werden, es ist also nur das Partizip II beider Verbformen involviert.

Für Formen des Perfekts, Plusquamperfekts oder Futurs II, die für ein Verb mit «*sein*», für das andere jedoch mit «*haben*» gebildet werden, wie «*führen*» (mit «*haben*») und «*werden*» (mit «*sein*»), ergeben sich jedoch weiterreichende Ersetzungen. Betrachten wir als Beispiel den Satz «*Im Bundestag* ***hat*** *das Thema zu einem heftigen Schlagabtausch* ***geführt***.», in dem wir «*führen*» durch «*werden*» ersetzen möchten. Das Resultat dieser Ersetzung sollte dann sein: «*Im Bundestag* ***ist*** *das Thema zu einem heftigen Schlagabtausch* ***geworden***.» Hier müssen auch die entsprechenden Vorkommen der Hilfsverben gefunden und ersetzt werden.

Komplexer wird der Austausch durch den Einbezug von Verbgefügen. Verbgefüge werden in infiniten Formen immer zusammengeschrieben. Die Formen des Präsens und Präteritums werden in Nebensätzen ebenfalls immer zusammengeschrieben, in Hauptsätzen ist jedoch die Form der Verbbasis das zweite Satzglied und der Verbzusatz steht an letzter Stelle. Für Ersetzungen unter Einbezug von Verbgefügen lassen sich somit vier Fälle unterscheiden: (a) Beide Verben sind kein Verbgefüge. (b) Beide Verben sind Verbgefüge. (c) Das ursprüngliche Verb ist ein Verbgefüge, das Ersatzverb ist keines. (d) Das ursprüngliche Verb ist kein Verbgefüge, jedoch das Ersatzverb.

Für die ersten beiden Fälle sind die Verbformen des ursprünglichen Verbs und des Ersatzverbs symmetrisch. Fall (a) haben wir bereits oben besprochen. Für (b) muss in Fällen, in denen Verbgefüge getrennt geschrieben werden, der ursprüngliche Verbzusatz durch den Verbzusatz des Ersatzverbgefüges ersetzt werden. Für (c) muss der Verbzusatz des ursprünglichen Verbgefüges entfernt werden. Diese drei Fälle sind relativ einfach.

Für (d) muss jedoch eventuell der Verbzusatz des Ersatzverbgefüges eingefügt werden. Hierzu ist es notwendig zu bestimmen, ob der aktuelle Satz ein Haupt- oder Nebensatz ist, um diese Entscheidung zu treffen und den Ort der Einfügung des Verbzusatzes zu bestimmen. Hierfür können wir pragmatische Lösungen unter Verwendung der bisher eingesetzten Ressourcen finden. Für alle Varianten kommt für Perfekt und Plusquamperfekt noch die Problematik der Hilfsverben «*sein*» und «*haben*» hinzu, wie obenschon gezeigt.

Wir können die Komplexität des Ersetzens anhand einer Matrix bestimmen, siehe Abbildung 7.17 auf der nächsten Seite. Liegen sowohl Originalverb *O* wie auch Ersatzverb *E* im gleichen Quadranten, ist die Ersetzung am unkompliziertesten, die Verbformen sind symmetrisch. Liegen *O* und *E* in zwei Quadranten, die diagonal zueinander stehen – also im 1. und 3. oder im 2. und 4. –, ist die Ersetzung am komplexesten. Liegen *O* und *E* in zwei nebeneinanderliegenden Quadranten – im 1. und 2., im 2. und 3, im 3. und 4. oder im 4. und 1. –, ist die Komplexität als «mittel» einzustufen.

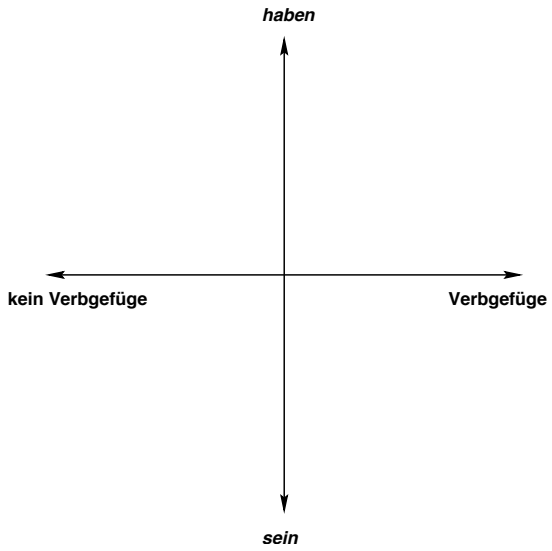


Abbildung 7.17: Komplexität von Verbersetzungen.

Alternativ können wir auch das Skalare Produkt der Vektoren verwenden. Dabei wird sowohl das zu ersetzende als auch das Ersatzverb durch einen Vektor beschrieben, der die Ausprägung bezüglich des Hilfsverbs («*sein*» entspricht -1, «*haben*» entspricht 1) und der Eigenschaft enthält, ein Verbgefüge (1) oder kein Verbgefüge (-1) zu sein. Wird etwa «*lesen*» (Hilfsverb «*haben*», kein Verbgefüge) durch «*schreiben*» (Hilfsverb «*haben*», kein Verbgefüge) ersetzt – wir haben also zwei Verben mit den gleichen Eigenschaften, die im gleichen Quadranten entsprechend Abbildung 7.17 liegen–, ist die Komplexität:

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \times \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 2$$

Werden zwei Verben mit völlig unterschiedlichen Eigenschaften getauscht, etwa «*gehen*» (mit «*sein*», kein Verbgefüge) mit «*überschnappen*» (mit «*haben*», Verbgefüge) – liegen die beiden Verben also in diagonal zueinander stehenden Quadranten entsprechend Abbildung 7.17 –, ist die Komplexität:

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \times \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix} = -2$$

Für Verben mit ähnlichen Eigenschaften (beide haben das gleiche Hilfsverb, jedoch ist eines ein Verbgefüge und das andere nicht; beide sind ein Verbgefüge bzw. beide sind keines und verwenden verschiedene Hilfsverben) – liegen die beiden Verben also in zwei benachbarten Quadranten entsprechend Abbildung 7.17 –, ist die Komplexität:

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} = 0$$

Im Hauptsatz steht das finite Verb an zweiter Stelle, in der Regel folgen zwischen finitem Verb und Satzende oder Interpunktionszeichen also noch weitere Wortformen. Dies lässt sich unter Verwendung der in *XEmacs* bereits vorhandenen Erkennung von *Satz* und *Wort* feststellen. In Nebensätzen steht die finite Verbform dagegen an letzter Stelle, also direkt vor dem Satzende oder einem Interpunktionszeichen. Falls der Satz vollständig ist, können wir dies also leicht feststellen.

(7.18) Rau **hatte** ferner *gesagt*, er **nehme** Fischer **ab**, dass ihm die damaligen Geschehnisse **leid täten**.

Im Beispielsatz 7.18 werden drei finite Verbformen gefunden: «*hatte*», «*nehme*» und «*täten*» (jeweils fett gedruckt sind die finiten Verbformen inklusive Verbzusätze und kursiv die zugehörigen infiniten Verbformen). «*täten*» steht direkt vor einem Interpunktionszeichen, hier handelt es sich also um einen Nebensatz. «*Hatte*» und «*nehme*» stehen nicht direkt vor einem Interpunktionszeichen, hier handelt es sich also um Hauptsätze.

(7.19) a. Vierzehn Forscher aus mehreren Staaten **sammelten** im UN-Auftrag im Kosovo Munitions- und Panzerreste.

b. Vierzehn Forscher aus mehreren Staaten **trugen** im UN-Auftrag im Kosovo Munitions- und Panzerreste **zusammen**.

Im Beispielsatz 7.19a finden wir einen Hauptsatz. Ersetzen wir die Form von «*sammeln*» durch die entsprechende Form von «*zusammentragen*», müsste das Resultat wie in Beispiel 7.19b lauten. Direkt vor dem Interpunktionszeichen wird also der Verbzusatz «*zusammen*» eingefügt.

Die gezeigten Besonderheiten machen deutlich, warum ein grosser Teil der von uns in Anhang A zusammengetragenen Fehler Verben betrifft, etwa «*Vom Seemannsgericht zur Spezialität: Labskaus sorgt erlebt ein Revival in den deutschen Küchen.*» (Beispiel A.134 auf Seite 279), «*Der Verband unabhängiger Schweizer Hochzeitsplaner (VSUH) bietet einen Diplomlehrgang zum Hochzeitsplaner durch.*» (Beispiel A.203 auf Seite 297). Beide Fehler hätten durch die Verwendung der von uns vorgeschlagenen Funktion vermieden werden können – im ersten Beispiel ist das zu ersetzende Verb weiterhin vorhanden, im zweiten Beispiel blieb der Verbzusatz «*durch*» (vermutlich von der ursprünglichen Version «*führte … durch*») erhalten.

7.3.5.1 Klassifizierung

Typ Modifikationsfunktion

Textveränderung ja

Spezifizität implizit über Funktionsaufruf, explizite Angabe des Ersatzverbs

Skopus Wort

Gebiet lokal (Satz)

linguistische Ressourcen Analyse von Wortformen mittels *GERTWOL*, Generierung von Wortformen mittels *GERGEN*, Ermittlung von Verbzusätzen mittels *Mbt* (dynamisch, Analyse und Generierung, morphologisch)

Seiteneffekte möglich, teils kontrollierbar

Eigener Modus electric mode

7.3.5.2 Funktionsweise, Beispiele

Der Autor platziert den Cursor auf dem Verb, das er durch ein anderes ersetzen möchte. Mit dem Aufruf von `replace-verb` werden alle Wortformen hervorgehoben, die durch den beabsichtigten Tausch innerhalb des aktuellen Satzes manipuliert werden können. Eine solche Wortform ist die, die durch die Cursor-Position bestimmt ist. Weitere Wortformen sind beteiligte Hilfsverben (d. h. Formen von «*sein*», «*haben*» und «*werden*») sowie Verbzusätze. Wir verwenden auch hier den *electric* Modus wie schon für `toggle-noungroup-number`.

Das Verb wird analysiert und im *electric* Buffer werden die Analysen angezeigt. Anschliessend werden die benötigten Wortformen für das Ersatzwort generiert. Im *electric* Buffer werden die Wortformen, die die markierten Wortformen im ursprünglichen Text ersetzen sollen, angezeigt. Dies schliesst auch Wortformen ein, die sich nicht ändern – etwa Hilfsverbformen, wenn eine Perfektform betroffen ist und beide Verben das Perfekt mit dem gleichen Hilfsverb bilden.

Für Formen, die nicht exakt bestimmt werden können, da die ursprüngliche Verbform ambig ist, wie für Formen, für die das Ersatzverb sowohl ein Verbgefüge als auch ein Kompositum sein kann, werden alle generierten Varianten angezeigt. Der Autor wählt mit den Cursor-Tasten und SPC die intendierte Form aus. Anschliessend werden die generierten Formen automatisch an der Stelle der ursprünglichen Formen eingefügt – Verbzusätze werden entfernt oder zusätzlich eingefügt – und der Modus wird beendet.

Verbgefüge werden in bestimmten Formen in Nebensätzen zusammen-, in Hauptsätzen jedoch getrennt geschrieben. Da wir keine tiefe syntaktische Analyse verwenden können, um Haupt- und Nebensätze voneinander zu unterscheiden, nutzen wir die Stellung des finiten Verbs. Wir haben gezeigt, dass es möglich ist zu entscheiden, dass ein Verb in einem Nebensatz verwendet wird, wenn direkt anschliessend ein Interpunktionszeichen folgt. Jedoch begegnen uns natürlich auch Beispiele wie 7.20a. Wollen wir «*fliehen*» durch «*weglaufen*» ersetzen, wird unsere Funktion ermitteln, dass «*weglaufen*» ein Verbgefüge ist und in einen Nebensatz eingefügt wird – direkt nach der ursprünglichen Verbform «*flieht*» folgt das Komma. Dies führt dazu, dass der in Beispiel 7.20b gezeigte Satz das Resultat der Ersetzung ist, tatsächlich würden wir jedoch den Satz 7.20c erwarten.

(7.20) a. Er **flieht**, weil er daheim keine Perspektive mehr sieht und in Ungarn das Grab seiner ehemaligen Frau sucht.

b. * Er **wegläuft**, weil er daheim keine Perspektive mehr sieht und in Ungarn das Grab seiner ehemaligen Frau sucht.

c. Er **läuft weg**, weil er daheim keine Perspektive mehr sieht und in Ungarn das Grab seiner ehemaligen Frau sucht.

Wir können uns hier mit Heuristiken behelfen, etwa dass ein Teilsatz aus lediglich zwei Wortformen, wobei die zweite das finite Verb ist, in der Regel ein Hauptsatz ist. Trotzdem liegt hier eine potenzielle Fehlerquelle. Entweder muss ein solcher Satz vom Autor nachbearbeitet werden, oder er muss vor dem Ersetzen die richtige Version auswählen. Hierbei kommt uns zugute, dass solche Zweifelsfälle nur dann auftreten, wenn Verbbasis und Verbzusatz direkt aufeinanderfolgen – befinden sich Wortformen zwischen Verbbasis und Verbzusatz,

handelt es sich in jedem Fall um einen Hauptsatz. Ist der Satz, in dem das Verb getauscht werden soll, noch nicht vollständig geschrieben, können wir keine Entscheidung treffen.

Für Konstruktionen mit einem Modalverb, wie in «*Die Plutonium-Spuren könnten einiges von dem erklären*» oder «*Das mag die Opposition erfreuen*», in denen das Modalverb erhalten werden soll und nur das Verb ausgetauscht wird, das den eigentlichen Inhalt transportiert, wird eine Infinitivform gegen eine andere Infinitivform ausgetauscht.

7.3.5.3 Implementierung

Listing 7.12: Pseudocode für die Funktion `replace-verb`.

```
switch to electric-verb-mode
start at point
highlight word at point
show-auxverb, highlight PTKVZ
get-analysis for finverbs and word at point
get-information-newverb
determine sentencetyp
create newverbform with category
allow choosing verbform
show modified sentence
allow choosing sentence
replace verbform
quit electric-verb-mode
```

Listing 7.12 zeigt die abstrakte Implementierung der Funktion. `show-finverb` ruft *Mbt* auf – dies kann eine kurze Verzögerung bedeuten – und hebt anschliessend die finite Verbform, infinite Verbformen und etwaige Verbzusätze (`PTKVZ`) hervor. Durch die Cursor-Position ist das zu ersetzende Verb bestimmt – dabei handelt es sich entweder um eine finite Verbform (für Formen des Präsens oder Präteritums) oder um eine infinite Verbform (für Formen des Perfekts, Plusquamperfekts, Futurs I oder II). Im Fall von infiniten Formen als zu ersetzendem Verb finden wir durch *Mbt* und die bereits beschriebene Funktion `show-finverb` (siehe Listing 7.5 auf Seite 201) die finite Form des Hilfsverbs – wir nennen diese Funktion `show-auxverb`.

Finite und infinite Verbformen werden anschliessend mittels `get-analysis` durch *GERTWOL* analysiert – falls noch kein Prozess für *GERTWOL*/*GERGEN* im Hintergrund läuft, ergibt sich hier nochmals eine kurze Verzögerung.[27] Damit wird die Kategorie der ursprünglichen Verbform ermittelt. Die Kategorie wird dann gemeinsam mit dem gewünschten Ersatzverb, das der Autor als Parameter für die Funktion angegeben hat, an *GERTWOL* übergeben. Das Ergebnis ist eine Wortformoberfläche des Ersatzwortes mit der Kategorie der ursprünglichen Wortform.

Für Verben im Präsens und Präteritum wird nur ein finites Verb gefunden und die entsprechende Verbform erzeugt. Für Verben im Perfekt und Plusquamperfekt finden wir für das ursprüngliche Verb jeweils eine finite Form von «*haben*» oder «*sein*» und eine Partizipform des Verbs. Wir müssen mittels *GERGEN* also die Partizipform für das Ersatzverb generieren.

Die Analyse der ursprünglichen Verbform (auf der der Cursor platziert ist) liefert keinen Hinweis darauf, ob Perfekt und Plusquamperfekt mit «*haben*»

27. Alternativ können sowohl *Mbt* als auch *GERTWOL*/*GERGEN* beim Start von *XEmacs* aufgerufen werden, sodass die Verzögerung bei der ersten Verwendung entfällt.

oder «*sein*» gebildet werden. Für diese Information können wir die Auszeichnung einer entsprechenden Form als finite Verbform durch *Mbt* verwenden. Jedoch benötigen wir eine entsprechende Information auch für das Ersatzwort. Eine Möglichkeit wäre, diese Information vom Autor über einen Parameter angeben zu lassen. Dies ist jedoch aus verschiedenen Gründen abzulehnen: Zum einen ist beim Aufruf der Funktion noch nicht ermittelt, ob es sich um die Ersetzung einer Perfekt- oder Plusquamperfektform handelt, ob diese Information also überhaupt benötigt wird. Zum anderen ist diese Information dem Autor möglicherweise bei der Angabe des Ersatzverbs nicht präsent, er müsste erst nachdenken – damit würden wir die kognitive Belastung jedoch unnötig erhöhen.

Wir gehen davon aus, dass der Autor nur die wirklich benötigten Informationen selbst explizit eingeben muss, die das System nicht selbst ermitteln kann – das zu ersetzende Verb wird etwa nur durch die Positionierung des Cursors bestimmt, der Autor gibt die Grundform nicht selbst ein. Dadurch reduzieren sich der kognitive Aufwand für den Autor und die Anzahl der auszuführenden Aktionen: Tastendrücke, Cursor-Positionierungen, Handbewegungen von der Tastatur zur Maus und umgekehrt, mentale Prozesse zur Vorbereitung auf die nächste Aktion (siehe [Raskin 2000, S. 73f]). Die Anzahl der notwendigen Aktionen halten wir möglichst klein, um die informationstheoretische Effizienz (*information-theoretic efficiency*) möglichst hoch zu halten:

> The **information-theoretic efficiency E** of an interface is defined as the minimum amount of information necessary to do a task, divided by the amount of information that has to be supplied by the user. [Raskin 2000, S. 84][28]

Von den verfügbaren Ressourcen zur morphologischen Analyse liefert lediglich *Stripey Zebra* bei der Analyse einer Verbform jeweils die Information zum Hilfsverb, jedoch nur für die Partizip II-Formen. Um diese Information ohne Interaktion mit dem Autor zu erhalten, müssen wir also eine Kombination aus *GERTWOL* und *Stripey Zebra* verwenden, wie in Listing 7.13 gezeigt.

Listing 7.13: Pseudocode für die Funktion `get-information-verb`.

```
ifnot GERTWOL running
  start GERTWOL
ifnot ZEBRA running
  start ZEBRA
create PII as verb+V-PART-PERF (GERTWOL)
analyse PII (ZEBRA)
get auxiliar-info
get verbzusatz
```

Wir können mit *Stripey Zebra* den Verbzusatz exakt ermitteln; *GERTWOL* weist hier Schwächen auf, wie in Abschnitt 6.2.4.3 gezeigt.[29] Wir verwenden daher die Funktion `get-information-verb` bei jedem Funktionsaufruf. In der Analyse von Partizipien durch *Stripey Zebra* ist die Information zum Hilfsverb im Attribut `PII_Auxiliary` zu finden.[30] Ein Verbzusatz wird in der Segmen-

28. Hervorhebung im Original.
29. Komplexe Verbzusätze wie «*hinauf*» in «*hinauftragen*» werden nicht korrekt erkannt.
30. Dieses Attribut wird nur für Partizipien ausgegeben, ist aber vermutlich im Lexikon notiert. Da wir leider nicht auf Lexikon und Regeln zugreifen können, können wir die Ausgabe nicht dahingehend ändern, dass dieses Attribut auch bei der Analyse des Infinitivs angegeben wird.

tierung durch die Segmentierungsmarke `<PFX>` im Attribut `Segmentation` gekennzeichnet.

Für Verben wie «*überlegen*», die sowohl ein Kompositum als auch ein Verbgefüge sein können, generieren wir mit *GERTWOL* zwei Partizipien: «*übergelegt*» und «*überlegt*». Die Analyse mit *Stripey Zebra* liefert dann die Resultate in den Abbildungen 7.18 und 7.19.

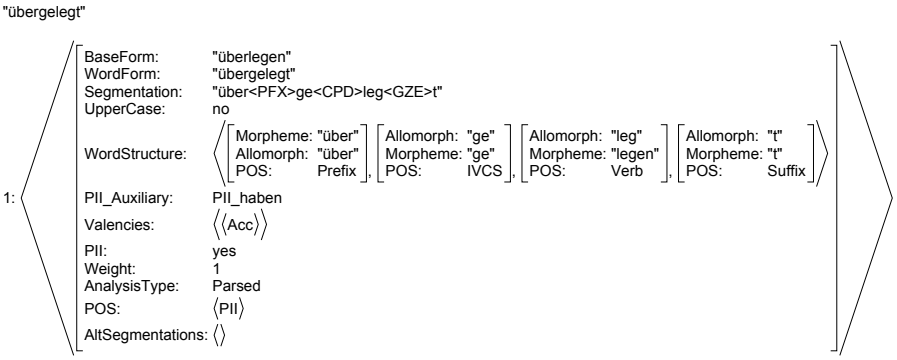


Abbildung 7.18: Analyse von «übergelegt».

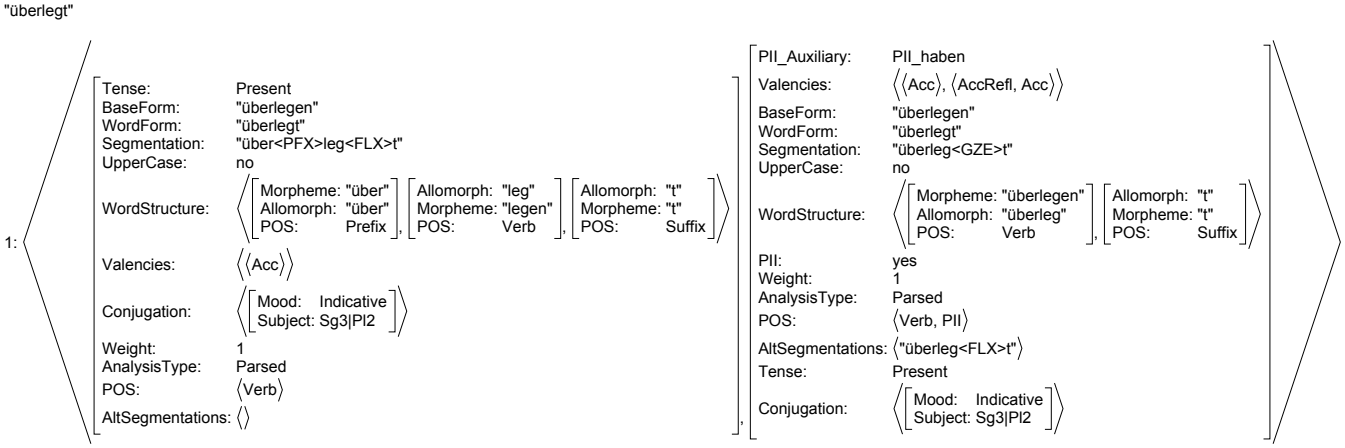


Abbildung 7.19: Analyse von «überlegt».

Leider werden von *GERTWOL* nach Angabe der Infinitivform und der gewünschten Kategorie nicht beide Partizipformen generiert. Die Form «*übergelegt*» erhalten wir nur, wenn wir die Information mitgeben, dass es sich um ein Verbgefüge handelt. Diese Information wiederum erhalten wir nur, wenn wir die Infinitivform mit *GERTWOL* analysieren.[31] *GERTWOL* erzeugt für Verbgefüge zudem nur die zusammengeschriebenen Varianten, die in Nebensätzen verwendet werden. Für die Verwendung in Hauptsätzen müssen wir jeweils die getrenntgeschriebene Variante erzeugen. Den Verbzusatz selbst haben wir mit Hilfe von *Stripey Zebra* bereits ermittelt, von der von *GERTWOL* generierten Verbform trennen wir diesen ab und erhalten die Verbbasis.

Ist für ein Ersatzverb das ermittelte Hilfsverb für die Verwendung mit dem Partizip II ein anderes als das von *Mbt* gefundene, muss auch das Hilfsverb ausgetauscht werden. In diesem Fall verwenden wir also das schon generierte Partizip II des Ersatzverbs und generieren zusätzlich die benötigte Form des Hilfsverbs.

31. Umso ärgerlicher ist der Fehler, der nur in der aktuellen Version von *GERTWOL* auftritt: Verben, die Verbbasis für ein Verbgefüge sein *können*, werden ebenfalls als Verbgefüge analysiert, siehe Abschnitt 6.2.6.3.

Im Gegensatz zum Ändern des Numerus einer Substantivgruppe trennen wir hier die Erzeugung der neuen Oberfläche und die Manipulation des Textes und führen sie jeweils in einem eigenen Schritt aus, da wir die oben schon genannten vier Varianten unterscheiden und eventuell zusätzlich das Hilfsverb austauschen müssen.

Nach dem Starten der Funktion sind alle relevanten Wortformen hervorgehoben, also etwaige Hilfsverben, die tatsächlich vorhandene finite oder infinite Form des Verbs und etwaige Verbzusätze. Nach der Analyse dieser Wortformen und der Generierung der benötigten Ersatzwortformen werden diese dem Autor präsentiert und er wählt die gewünschte Ersetzung. Für Ersatzverben bzw. -verbgefüge wie «*überlegen*» werden je nach syntaktischem Kontext mehrere Ersatzmöglichkeiten präsentiert. Der Autor wählt die gewünschte Variante des Satzes. In beiden Fällen kann die Ersetzung auch abgebrochen werden.

Um einen etwaigen Verbzusatz korrekt zu platzieren oder die zusammengeschriebene Form zu wählen, bestimmen wir, ob sich das zu ersetzende Verb in einem Haupt- oder Nebensatz befindet. Dazu ermitteln wir, ob der finiten Verbform direkt das Interpunktionszeichen folgt – dann ist es ein Nebensatz – oder nicht – dann ist es ein Hauptsatz.

7.3.5.4 Seiteneffekte, Folgefunktionen

Wir haben oben schon auf verschiedene Faktoren hingewiesen, die das Ersetzen von Verben zu einer komplexen Aufgabe machen. Im Beispielsatz 7.21 finden wir einen Haupt- und einen Nebensatz. Wird das Verb des Nebensatzes («*liefert*») durch «*andeuten*» ersetzt, wird die entsprechende Form zusammengeschrieben. Im Hauptsatz handelt es sich um eine Verbform von «*stossen*» mit Hilfsverb. Eine Ersetzung gestaltet sich hier unter Umständen schwierig: Eigentlich handelt es sich um eine Form mit Hilfsverb «*sein*» von «*auf etwas stossen*». Möchten wir dies durch «*finden*» ersetzen, muss das Hilfsverb getauscht werden («*sind*» gegen «*haben*»). Zudem müssten wir die Präposition «*auf*» entfernen, da die korrekte Formulierung wäre «*Sie haben nun einen Stoff gefunden, …*»

(7.21) Sie **sind** nun auf einen Stoff *gestoßen*, der deutliche Hinweise auf Plutonium **liefert**: Uran-236.

Platziert der Autor den Cursor auf «*gestossen*», besteht keine Möglichkeit, automatisch festzustellen, dass es sich um die Formulierung «*auf etwas stossen*» handelt, die hier gegen «*finden*» getauscht werden soll. Da die beteiligten Wortformen («*auf*» und «*gestossen*») nicht direkt aufeinanderfolgen, besteht ebenso keine Möglichkeit, dass der Autor sie vor dem Aufruf der Funktion gemeinsam mittels Maus oder Tastatur markieren könnte. Diese Ersetzungsoperation können wir hier also nicht wesentlich erleichtern, der Autor muss anschliessend den Satz prüfen (oder prüfen lassen) und vermutlich korrigieren.

Ähnlich verhält es sich mit Fällen, (a) in denen ein transitives durch ein intransitives Verb ersetzt wird oder umgekehrt (einige Verben können zudem sowohl transitiv als auch intransitiv gebraucht werden, etwa «*geben*»), (b) in denen reflexive Verben beteiligt sind (einige Veben können sowohl reflexiv als auch nichtreflexiv gebraucht werden, etwa «*waschen*») oder (c) in denen die Transitivität identisch bleibt, jedoch das Ersatzverb einen anderen Kasus

regiert als das ursprüngliche. Wie schon für die Bestimmung von Haupt- und Nebensätzen wären wir für eine exakte Bestimmung darauf angewiesen, dass der bearbeitete Satz bereits vollständig geschrieben ist – und dass er vor der Manipulation auch syntaktisch wohlgeformt war. Von beiden Gegebenheiten können wir nicht ausgehen und müssen daher eine pragmatische Lösung finden.

GERTWOL liefert keine Informationen über Reflexivität oder Transitivität[32], ebenso können wir mit den uns zur Verfügung stehenden morphologischen Systemen keine Angaben zu kollokativ gebrauchten Verben wie «*auf etwas stossen*» erhalten. Das ist kein Mangel dieser Systeme, da diese Angaben nicht zu den morphosyntaktischen Eigenschaften einer Wortform gehören, sondern etwas über ihren syntaktischen oder gar semantisch-pragmatischen Einsatz aussagen, also von anderen Systemen ermittelt werden müssten. Lediglich *Stripey Zebra* enthält im Attribut `Valencies` Angaben zum Valenzrahmen des analysierten Verbs. Wollten wir diese Information verwenden, müssten wir also in der Funktion `get-information-verb`, in der wir für das Ersatzverb das Hilfsverb («*sein*» oder «*haben*») sowie den allfälligen Verbzusatz ermitteln, zusätzlich diese Information extrahieren. Wir benötigen diese Information jedoch ebenfalls für das ursprüngliche Verb. Ein zusätzlicher Funktionsaufruf verlängert insgesamt die Ausführungszeit der Funktion; abhängig von den extrahierten Eigenschaften der beteiligten Verben müssten zusätzlich Interaktion mit dem Autor angefordert oder bestimmte weitere Einfügungen oder Löschungen vorgenommen werden. Insgesamt würde dies den gesamten Prozess extrem verkomplizieren und vor allem verlangsamen.

Wir schlagen darum vor, dass die geänderten Textstellen *nach* dem Beenden der Funktion hervorgehoben bleiben – der Autor sieht explizit die Auswirkungen der Funktion, kann nun aber selbst korrigieren durch Eingabe oder Löschen von Text oder durch den Aufruf anderer Funktionen. Erst nach dem Drücken der ersten Taste werden die Markierungen gelöscht. Zudem kann der Autor eine Informationsfunktion zur Hervorhebung von Textstellen verwenden, die zuletzt bearbeitet wurden (siehe Abschnitt 4.2).

Zusammenfassend lässt sich feststellen, dass wir hier an die Grenzen der Modifikationsfunktionen gelangen – einerseits aus computerlinguistischer Sicht, andererseits aus der Perspektive der Nützlichkeit. Die heute verfügbaren computerlinguistischen Ressourcen können die benötigten Informationen nicht schnell und vor allem nicht korrekt genug bereitstellen, um alle Aspekte einer Operation zur Ersetzung von Verben zu berücksichtigen. Zudem steht die Möglichkeit der tiefen syntaktischen Analyse nicht zur Verfügung, da wir nicht fertigen wohlgeformten Text bearbeiten, sondern Text, der erst entsteht. Es existieren leider keine systematischen Untersuchungen zu morphosyntaktischen Eigenschaften von Verben, die in bestimmten Beziehungen zueinander stehen. Zudem liegen keine linguistisch motivierten Untersuchungen der Textproduktion vor. Die Log-Daten von Autoren werden bislang nicht intensiv auf linguistischer Ebene ausgewertet, wir können daher nicht ermitteln, ob bei der Ersetzung von Wörtern oder Wortgruppen eher synonym gebrauchte Wörter oder Wortgruppen verwendet werden oder eher antonym gebrauchte. Diese Information wäre notwendig, um für Operationen wie das Ersetzen von Verben die computerlinguistischen Ressourcen optimal einsetzen zu können.

32. Haapalainen und Majorin [1994] kündigen zwar Erweiterungen bezüglich Valenzangaben an, diese wurden jedoch nie umgesetzt.

Aus computerlinguistischer und eher theoretischer Sicht mag eine Beschäftigung mit den prinzipiell möglichen Varianten bei der Ersetzung von Verben interessant sein. Ob eine Berücksichtigung aller denkbaren Varianten in der Implementierung jedoch notwendig ist, kann ebenfalls erst nach einer intensiven Beschäftigung mit den Daten der Schreibforschung festgestellt werden. Die computerlinguistischen Ressourcen sind nie vollständig zuverlässig und liefern nie vollständig korrekte Resultate. Da wir hier also Abstriche machen müssen, könnten wir Varianten der Verbersetzung, die nur sehr selten vorkommen, explizit von der automatischen Behandlung ausschliessen, ohne die allgemeine Nützlichkeit einer solchen Funktion zu verringern.

Der Hauptzweck der von uns implementierten Funktionen ist nicht das Ausreizen des technisch Machbaren, sondern die Reduktion der kognitiven Belastung für Autoren. Wir sind daher nicht interessiert an Funktionen, die zu einem bestimmten zu ermittelnden Prozentsatz korrekte Resultate liefern, sondern an Funktionen, die möglichst immer korrekte Resultate liefern und dem Autor händisches Editieren ersparen. Wir kommen noch einmal auf die bereits in Abschnitt 4.4 genannte Quote von mehr als 90 % korrekter Resultate zurück – die Resultate der Funktionen müssen verlässlich sein, ansonsten würde die kognitive Belastung durch eine zusätzlich notwendige Inspektion aller Resultate unnötig erhöht. Die Nützlichkeit der Funktion wäre in Frage gestellt.

7.3.6 Zusammenfassung: Implementierung konkreter Funktionen

In diesem Abschnitt haben wir Implementierungen konkreter Funktionen vorgestellt und Beispiele für Informations-, Bewegungs- und Modifikationsfunktionen gezeigt. Wir benutzen den in Abschnitt 4.4.1 erläuterten Grundsatz, vorhandene Funktionen zu kombinieren, um den Implementierungsaufwand und mögliche Fehlerquellen zu reduzieren. So lässt sich auch sicherstellen, dass bei der Verwendung einer anderen computerlinguistischen Ressource – etwa für die morphologische Analyse und Generierung von Wortformen – lediglich eine Anpassung der Schnittstelle zu dieser Ressource notwendig ist, um den hier vorgestellten Funktionsumfang beizubehalten.

Für die vorgestellten Funktionen haben wir jeweils die benötigten Ressourcen benannt. Die Implementierung zeigt die generelle Umsetzbarkeit des Konzepts linguistisch unterstützter Funktionen. Wir haben jeweils auch die Grenzen des Funktionsumfangs aufgezeigt. Je mehr Ressourcen miteinander kombiniert werden müssen, desto grösser wird die Fehlerrate. Für Funktionen, die von syntaktischen Informationen abhängig sind oder sich auf sehr variable Elemente, wie Verben, beziehen, stellt sich dann die Frage nach der Nützlichkeit, da u. U. umfangreiche Interaktion mit dem Benutzer nötig ist. In den folgenden Abschnitten zeigen wir, welche weiteren Funktionen sich aus den bereits implementierten ableiten lassen.

7.4 Ableitung weiterer Funktionen

Alle vorgestellten Funktionen haben eine spezifische Ausprägung entsprechend der Klassifikation, die wir in Abschnitt 4.3 vorgestellt haben. Wir können für jede Funktion ähnliche Funktionen ableiten, indem wir jeweils Skopus

und / oder Gebiet variieren. In den folgenden Abschnitten gehen wir genauer auf Varianten der drei Funktionstypen ein.

7.4.1 Informationsfunktionen

Die von uns vorgestellte Informationsfunktion `show-sentence-without-finverb` (Abschnitt 7.3.1) arbeitet global und bezieht sich auf Wörter und Sätze.

Variieren wir nur das Gebiet, haben wir eine Funktion, die für den aktuellen Satz ermittelt, ob ein finites Verb vorhanden ist.

Betrachten wir den Skopus, ergeben sich mehrere Varianten. In der vorgestellten Funktion wurden Sätze *ohne* finites Verb hervorgehoben, wir können aber natürlich auch nach Sätzen *mit* finiten Verben suchen. Zudem lässt sich die Wortart variieren, etwa in der Suche nach Sätzen ohne Adverb. Wir können den Skopus auch anpassen und nach Sätzen mit oder ohne Substantivgruppen allgemein oder nach Sätzen mit / ohne Substantivgruppen einer bestimmten Kategorie suchen. So könnten beispielsweise Sätze ohne explizites Subjekt gefunden werden. Eine andere Variation des Skopus betrifft etwa die Hervorhebung von Wortgruppen mit / ohne Wortformen einer bestimmten Wortart. Alle Variationen können als globale und als lokale hervorhebende Informationsfunktion implementiert werden.

Erweitern wir die Informationsfunktion, sodass sie nicht nur gefundene Elemente hervorhebt, erhalten wir Informationsfunktionen mit Rückgabewert, die für globale Funktionen die Anzahl der Fundstellen zurückgeben. Wird nach Wortgruppen gesucht, kann der Rückgabewert der Typ der Wortgruppe und / oder die Kategorie sein.

7.4.2 Bewegungsfunktionen

Varianten für Bewegungsfunktionen können wir ebenfalls – ähnlich wie wir es oben in Abschnitt 7.4.1 gezeigt haben – von der bereits implementierten Funktion, nämlich `goto-next-finverb` (Abschnitt 7.3.2), ableiten, indem wir den Skopus variieren. Zudem kann für jede solche Funktion das Ziel variiert werden, indem entweder zum vorigen oder zum nächsten Element navigiert wird. Für längere Ziele, etwa Wortgruppen, kann zudem zum Beginn oder zum Ende des Ziels navigiert werden.

Eine andere Ableitung von möglichen Bewegungsfunktionen ergibt sich direkt aus den Informationsfunktionen. Jedes Element, das durch eine Informationsfunktion hervorgehoben wird, kommt als Ziel einer entsprechenden Bewegungsfunktion in Frage. Auch hierbei können wir jeweils zum vorigen oder nächsten Element und zum Beginn oder zum Ende eines solchen Elements navigieren.

Für alle Bewegungsfunktionen ist die Kombination mit einer entsprechenden Informationsfunktion sinnvoll: Das Navigationsziel wird nach der Positionierung des Cursors hervorgehoben, um eine schnelle Orientierung zur ermöglichen.

7.4.3 Modifikationsfunktionen

Wie für Informations- und Bewegungsfunktionen können wir durch Änderung von Gebiet und Skopus aus den vorgestellten Modifikationsfunktionen weitere ableiten. Wir haben drei Modifikationsfunktionen implementiert, nämlich `transpose-conjuncts`, `toggle-noungroup-number` und `replace-verb`.

Alle drei sind lokale Funktionen; `toggle-noungroup-number` und `replace-verb` könnten leicht in eine globale Funktion geändert werden.

Wir haben mit diesen Funktionen bereits Varianten für Skopus und die Art der Textänderung gezeigt. Modifikationsfunktionen können geschriebene Elemente hinsichtlich verschiedener morphosyntaktischer Eigenschaften verändern (`toggle-noungroup-number`). Sie können auch zum Ersetzen eines Elements durch ein anderes unter Beibehaltung der morphosyntaktischen Eigenschaften verwendet werden (`replace-verb`) oder zum Löschen oder Verschieben eines Elements oder einer Struktur. Für jede dieser prinzipiellen Möglichkeiten können wir den Skopus so variieren, dass verschiedene Wortarten oder Typen von Wortgruppen betroffen sind.

Die Fehlerrate solcher globalen Funktionen ist aufgrund der verwendeten computerlinguistischen Ressourcen und der schon beschriebenen Grenzen für die lokale Version dieser Funktionen voraussagbar hoch, sodass die Interaktion mit dem Benutzer in sehr vielen Fällen notwendig ist. Globalität stellt erhöhte Ansprüche an Gebrauchstauglichkeit und Bedienfreundlichkeit. Es werden Textstellen bearbeitet, die der Autor im Moment des Aufrufs der Funktion nicht sieht, er kann also nicht einschätzen, wo sich eine Änderung manifestieren wird – zudem kann er die Korrektheit dieser Änderung nicht voraussehen und darum nicht beurteilen, ob eine Interaktion notwendig sein mag. Ist eine Interaktion notwendig, wird sich der sichtbare Textausschnitt so verändern müssen, dass die fragliche Textstelle sichtbar ist – der Autor kann nicht einschätzen, wie weit er sich von der Textstelle, an der er die Funktion aufgerufen hat, entfernt hat und wie er wieder dorthin zurückgelangt, wenn er die Ausführung der Funktion hier unterbricht. Zudem ist eventuell gewünscht, nach der vollständigen Ausführung aller Änderungen – wenn die Funktion also nicht unterbrochen wird – alle Textstellen, die geändert wurden, zu inspizieren. Prinzipiell ist dies möglich, da diese etwa vor Verlassen eines Modus hervorgehoben werden können. Vorstellbar ist auch, eine Informationsfunktion zu verwenden, die kürzlich veränderte Textstellen hervorhebt. Auch hier bewegt sich der Autor jedoch von der ursprünglichen Textstelle weg.

Für globale Operationen muss also entweder ein anderes Bedienkonzept verwendet werden oder sogar aufgrund der Qualität der benötigten computerlinguistischen Ressourcen momentan auf die Implementierung verzichtet werden. Allenfalls ist ihre Verwendung in einem expliziten Redigiermodus sinnvoll, wobei sich der Benutzer im Klaren ist, dass er stark interagieren muss – aus angebotenen Varianten auswählen, Parameter nachträglich spezifizieren, hervorgehobene Textstellen anschliessend selbst noch einmal kontrollieren und gegebenenfalls ändern.

7.5 Zusammenfassung

In diesem Kapitel haben wir exemplarische Funktionen vorgestellt, die zeigen, dass das Konzept des linguistisch unterstützten Redigierens (s. Abschnitt 4.2) unter Verwendung der in Kapitel 6 ausgewählten computerlinguistischen Ressourcen implementiert werden kann. Für jeden der drei Funktionstypen (Informations-, Bewegungs- und Modifikationsfunktionen) haben wir konkrete Funktionen ausführlich beschrieben und sind auf Implementierungsdetails eingegangen. Aus diesen Funktionen können durch Variation der Eigenschaften von Gebiet und Skopus (siehe Klassifizierung in Abschnitt 4.3) weitere Funktionen abgeleitet werden.

Insbesondere für Modifikationsfunktionen haben wir auch die Grenzen der heutigen Möglichkeiten gezeigt. Wir orientieren uns zwar an den konkreten Bedürfnissen von Autoren und nicht am technisch Vorstellbaren, dennoch sind die Möglichkeiten klar durch die Qualität und Verfügbarkeit computerlinguistischer Ressourcen eingeschränkt.

8 Zusammenfassung, Diskussion und Ausblick

In short, we each seek, if not use, what we consider to be the word-processing programm which best facilitates, if not emulates, the easy eruption of words and which interferes the least with the brain-to-hand-to-keyboard-to-screen connection.

—Lee Roger Taylor, Jr., *Software Views: A Fistful of Word-Processing Programs*, 1986

Researchers have found that children and adults of different backgrounds can master the communication skills needed for interacting with a text editor.

—Colette A. Daiute, *The Computer as Stylus and Audience*, 1983

In dieser Arbeit haben wir uns damit beschäftigt, wie typische Redigierfehler beim Schreiben mit einem Textbearbeitungsprogramm vermieden werden können. Wir haben dazu das Konzept des linguistisch unterstützten Redigierens entwickelt und ausgewählte Funktionen implementiert.

Im Folgenden fassen wir die vorliegende Arbeit zusammen, diskutieren unseren Ansatz und geben einen Ausblick auf weitere Forschungsfragen.

8.1 Zusammenfassung

In natürlichsprachlichen Texten finden wir Fehler, deren Entstehung wir oft rekonstruieren und auf unvollständig ausgeführte Redigieroperationen zurückführen können. Solche Fehler lassen sich als *slips* nach Norman [1981] kategorisieren. Generell entstehen *slips*, wenn die Bedienoberfläche eines Werkzeugs den Benutzer nicht optimal unterstützt. Wir haben dies erstmals auf Textbearbeitungsprogramme und Fehler in natürlichsprachlichen Texten übertragen.

Studien der Schreibforschung belegen, dass Redigieren als ein Teilprozess des Schreibens kognitiv anspruchsvoll ist. Wir haben uns auf das Redigieren konzentriert, da die innerhalb dieses Prozesses auftretenden Schwierigkeiten für die beobachtbaren Fehler ursächlich sind, wie Sätze ohne finites Verb, falsche Wortstellung, Wortwiederholungen oder fehlende Kongruenz. Redigieren ist die Veränderung des bereits Geschriebenen (oder sogar des noch zu Schreibenden) zu jedem beliebigen Zeitpunkt und nicht primär das Finden und Korrigieren von Fehlern. Gründe für eine Änderung sind Erkenntnisse des Autors während des Schreibens, veränderte rhetorische Absichten oder das Ausprobieren verschiedener Formulierungen. Autoren sollten hierbei durch das verwendete Werkzeug – ihr Textbearbeitungsprogramm – optimal unterstützt werden.

Funktionen in Textbearbeitungsprogrammen operieren jedoch auf Zeichen, nicht auf linguistischen Elementen und Strukturen. Für die Umsetzung einer Redigierabsicht muss der Autor daher die gewünschte Operation, die sich auf die Manipulation linguistischer Elemente und Strukturen bezieht, in eine lange, komplexe Folge zeichenbasierter Funktionen übersetzen. Zum einen ist dieser Übersetzungsprozess selbst fehleranfällig, zum anderen ist die Ausführung solcher Folgen von Funktionen anfällig für *slips*. Wäre es möglich, auf grösseren – linguistisch motivierten – Einheiten zu operieren, würden beide Fehlerquellen minimiert.

In Texteditoren finden wir seit langem sprachbasierte Funktionen, die auf den Elementen und Strukturen der verwendeten Programmier- oder Auszeichnungssprache operieren. Solche Funktionen entlasten den Programmierer / Autor und dienen explizit der Fehlervermeidung. Sprachbasierte Funktionen in Texteditoren verwenden Hilfsprogramme, die die syntaktische Struktur des bereits Geschriebenen analysieren.

In dieser Arbeit haben wir dieses Prinzip erstmals systematisch auf Funktionen in Textbearbeitungsprogrammen übertragen und damit eine Forschungslücke geschlossen. Entsprechend den Erkenntnisse der Schreibforschung zum Redigierprozess und unter Berücksichtigung der Designprinzipien für sprachbasierte Funktionen in Texteditoren beruht das Konzept des linguistisch unterstützten Redigierens auf der Kombination bereits vorhandener Funktionen und dem Einsatz von Ressourcen zur flachen Analyse natürlichsprachlicher Texte.

Dafür verwenden wir computerlinguistische Ressourcen für Deutsch, die wir systematisch evaluiert haben. Die Umsetzbarkeit unseres Ansatzes haben wir mit der exemplarischen Implementierung von Funktionen gezeigt. Im folgenden Abschnitt 8.2 stellen wir dar, welche Erkenntnisse wir in Bezug auf die Hypothesen dieser Arbeit gewonnen haben.

8.2 Hypothesen

In Abschnitt 1.2 haben wir zwei Hypothesen aufgestellt, die wir mit dieser Arbeit untersuchen wollten:

H1: Analog zu sprachbasierten Funktionen in Editoren für Programmiersprachen ist es möglich, sprachbasierte Funktionen für Textbearbeitungsprogramme zu implementieren, die auf den strukturellen Einheiten der verwendeten natürlichen Sprache operieren.

H2: Linguistisch unterstützte Funktionen können unter Verwendung einfacher computerlinguistischer Ressourcen implementiert werden.

In Abschnitt 4.1.1 haben wir sprachbasierte Funktionen in Texteditoren vorgestellt und deren Prinzipien erörtert. Sprachbasierte Funktionen beruhen auf der Analyse der syntaktischen Elemente des bereits Geschriebenen entsprechend den Regeln der verwendeten Sprache. Solche Funktionen in Texteditoren können in drei Funktionsklassen eingeordnet werden: Informationsfunktionen, Bewegungsfunktionen und Modifikationsfunktionen. Aufgrund der Ähnlichkeit von natürlichen und formalen Sprachen können wir auch für die Bearbeitung von natürlichsprachlichen Texten sprachbasierte Funktionen konzipieren. Wir haben diese als *linguistisch unterstützte Redigierfunktionen* bezeichnet und in Abschnitt 4.2 vorgestellt.

Die erste Hypothese stützt sich auf eine Annahme, die wir in Abschnitt 1.2 genannt haben:

A2: Die Schreibforschung liefert empirische Befunde, welche Redigieroperationen aufgrund der verwendeten Funktionen in Textbearbeitungsprogrammen besonders anfällig für *slips* sind.

In Kapitel 2 haben wir Erkenntnisse aus der Schreibforschung zum Redigieren vorgestellt. Diese berücksichtigen das verwendete Werkzeug jedoch nicht, sondern sind auf Beobachtungen an der Textoberfläche beschränkt. In Kapitel 5 haben wir Untersuchungen zum Verhältnis von Schreibwerkzeug und Schreibprozess erörtert. Wir mussten feststellen, dass die Schreibforschung momentan die von uns erhofften Daten nicht bereitstellen kann, da die Beobachtung von Schreibprozessen nicht konkrete Funktionen von Schreibwerkzeugen, sondern lediglich allgemeine Charakteristika berücksichtigt. Jedoch wurden im letzten Jahr verschiedene Projekte begonnen, die sich auch solchen Fragen widmen, sodass wir zu einem späteren Zeitpunkt die entsprechenden Erkenntnisse verwenden können. Für den Moment müssen wir uns auf Selbstbeobachtung beschränken, wie viele der in Abschnitt 3.2 vorgestellten Projekte zur Schreibunterstützung.

In Kapitel 7 haben wir exemplarisch einige der mit der in Abschnitt 4.3 vorgestellten Klassifizierung beschreibbaren Funktionen implementiert. Die erste Hypothese können wir somit bestätigen.

Für die Implementierung haben wir computerlinguistische Ressourcen zur automatischen morphologischen Analyse und Generierung, zur automatischen

Wortartenbestimmung und zur Analyse von Wortgruppen verwendet. Diese Ressourcen liefern lediglich eine flache und keine tiefe syntaktische Analyse des Geschriebenen. In Abschnitt 4.4.1 haben wir gezeigt, dass die Verwendung tiefer syntaktischer Analyse nicht möglich ist: Automatische syntaktische Analyse geht von vollständigen wohlgeformten Sätzen aus. Während des Schreibens ist ein Text jedoch fast immer unvollständig und in Teilen ungrammatisch. Zudem zeigen Studien der Schreibforschung, dass Autoren einen Satz redigieren, *bevor* sie diesen vollständig geschrieben haben.

Insbesondere die zweite Hypothese stützt sich auf zwei Annahmen, die wir ebenfalls in Abschnitt 1.2 vorgestellt haben:

A1: Heutige Hardware kann rechenintensive Prozesse, wie sie die Anwendung computerlinguistischer Ressourcen darstellen, schnell genug ausführen, um solche Ressourcen interaktiv zu verwenden.

A3: Benötigte computerlinguistische Ressourcen für Deutsch sind in einer entsprechenden Qualität verfügbar, um sie in unbeschränkten Applikationen einzusetzen.

Die erste Annahme konnten wir bereits in Abschnitt 1.2 bestätigen. In Kapitel 6 haben wir verschiedene computerlinguistische Ressourcen für Deutsch evaluiert. Wir mussten feststellen, dass die zweite Annahme nicht gerechtfertigt ist. Weder sind Ressourcen für Deutsch frei verfügbar – insbesondere was automatische morphologische Analyse und Generierung betrifft –, noch ist die Qualität der Resultate solcher Systeme ausreichend, um in praxistauglichen Anwendungen eingesetzt zu werden. Wir haben seit der ersten Morpholympics 1994 die erste umfassendere Evaluation morphologischer Systeme für Deutsch durchgeführt. Die Ergebnisse sind ernüchternd. Die Korrektheit der Resultate erreicht nicht die allgemein postulierten 90 %, die für die Einbindung in praxistaugliche Applikationen vorausgesetzt werden. Annotierte Korpora, die als Ressource für statistische Verfahren dienen, berücksichtigen zudem nicht die aktuell verwendeten Rechtschreibregeln.

Aus den von uns evaluierten Systemen haben wir die am besten geeigneten ausgewählt und sie in den implementierten Funktionen verwendet. Berücksichtigt man die Fehler, die durch diese Ressourcen selbst verursacht werden, ist die Qualität der Resultate der implementierten Funktionen zufriedenstellend. Flache syntaktische Analyse sowie Analyse und Generierung von Wortformen sind ausreichend, um Informations-, Bewegungs- und Modifikationsfunktionen zu implementieren. Auch die zweite Hypothese können wir also bestätigen.

8.3 Diskussion

Die von uns vorgeschlagenen linguistisch unterstützten Funktionen zum Redigieren von natürlichsprachlichen Texten sind theoretisch breit abgestützt. Wir haben die Prinzipien sprachbasierter Funktionen in Texteditoren unter Berücksichtigung der Unterschiede von formalen und natürlichen Sprachen auf Funktionen zur Bearbeitung von natürlichsprachlichem Text übertragen und Erkenntnisse der Schreibforschung zum Redigierprozess einbezogen.

Einige Funktionen haben wir in Analogie zu bereits vorhandenen Funktionen im Editor *XEmacs* als *proof-of-concept* implementiert. Diese Implementierung entspricht durch die Anlehnung an bereits existierende Funktionen in *XEmacs* den Prinzipien der Bedienfreundlichkeit und Ergonomie, [vgl. ISO 1998, 2001]. Einige Eigenschaften sind jedoch klar von den verwendeten Ressourcen abhängig und werden durch deren Qualität beeinflusst, insbesondere Funktionalität (vor allem Korrektheit der Ergebnisse), Effizienz und Übertragbarkeit. Die Funktionen können in jede beliebige *XEmacs*-Instanz integriert werden, jedoch können nur Funktionen, die lediglich automatische Wortartenbestimmung mit *Mbt* verwenden, ohne Änderungen benutzt werden. Für Funktionen, die morphologische Analyse und Generierung mit *GERTWOL/GERGEN* oder *Stripey Zebra* verwenden, müssen erst entsprechende Lizenzen erworben oder andere Ressourcen benutzt werden. Es hat sich leider während der Evaluation herausgestellt, dass frei verfügbare morphologische Ressourcen nicht für praxistaugliche Anwendungen geeignet sind.

Komplexe Redigierziele wurden so operationalisiert, dass sie implementiert werden können. Die ausgewählten Redigieroperationen entsprechen Aktionen, die erfahrene Autoren beim Erstellen von deutschsprachigen wissenschaftlichen Texten vornehmen. Mit den bisher verfügbaren Werkzeugen können diese Redigierarbeiten ausgeführt werden, stellen jedoch hohe Anforderungen an den Autor, da eine Redigieroperation in eine lange, komplexe Sequenz kleiner Arbeitsschritte zerlegt werden muss. Durch die Implementierung von Funktionen, die es ermöglichen, ein komplexes Ziel mit *einem* Funktionsaufruf zu erreichen, hoffen wir, das Redigieren zu erleichtern und typische Fehler zu vermeiden.

Es interessiert daher, wie die intendierte Benutzergruppe (erfahrene Autoren[1]) die implementierten Funktionen annimmt. In Anlehnung an Babaian et al. [2002, S. 13] wurden verschiedenen Personen die Funktionen zur Verfügung gestellt und um Rückmeldung in Bezug auf folgende Aspekte gebeten: (a) Wie schwierig ist die Bedienung der Funktionen zu erlernen? (b) Für welche Texte und wie oft wird sie verwendet? (c) Wie nützlich ist die Funktion? (d) Wie wurden bisher die Ziele erreicht, die die Funktion ermöglicht? (e) Wird die Funktion in Zukunft genutzt? (f) Welche Veränderungen sind notwendig? Die Frage (d) war explizit notwendig, da wir, wie in Abschnitt 5.3 erörtert, nicht auf entsprechende empirische Daten oder Experimente der Schreibforschung zurückgreifen können. Hier wird also explizit nach der reflektierenden Selbstbeobachtung der Autoren gefragt. Es handelte sich um offene Fragen.

Beispielsweise wurde eine erste Version der Funktion `transpose-conjuncts` zwölf Personen zur Verfügung gestellt, von denen sechs unsere Fragen ausführlich beantworteten.[2] In dieser Version war es noch nicht möglich, Konjunkte auszuwählen, die nicht direkt an die Konjunktion anschliessen. Die Rückmeldungen bezogen sich vor allem darauf, nicht nur Wörter oder Wortgruppen tauschen zu können, die *direkt* an die Konjunktion anschliessen, sondern beispielsweise auch Köpfe von Substantivgruppen. In der Regel ist das Substantiv das am weitesten rechts stehende Wort der Wortgruppe. Damit schliesst der Kopf des linken Konjunkts direkt an die Konjunktion an, der Kopf des rechten Konjunkts jedoch nicht. Die Funktion wurde darum dahingehend erweitert,

1. «erfahren» sowohl im Hinblick auf das Verfassen von Texten der intendierten Textsorte als auch im Hinblick auf den Umgang mit *XEmacs*.
2. 50 % Rücklauf ist ein relativ hoher Wert.

die Konjunkte frei wortweise auswählen zu können, wie es in Abschnitt 7.3.3 gezeigt wird.

Die Lernaufwand wurde durchgängig als gering eingeschätzt, da die Funktion analog zu bereits existierenden Funktionen zu bedienen ist – eine Bestätigung des Designprinzips. Verwendet wurde die Funktion in verschiedenen Texten von E-Mail-Nachrichten bis zu Dissertationen und in verschiedenen Sprachen (deutsch und englisch). Eine Person verwendet die Funktion auch für Elemente, die nicht als Koordination anzusehen sind, die aber auch der Struktur `linkes Element–Mitte–rechtes Element` folgen und erklärte, sie dafür weiterhin benutzen zu wollen.

Die Frage nach der Nützlichkeit hängt eng mit der Frage zusammen, wie entsprechende Operationen bislang ausgeführt wurden. Eine Person erklärte, sie würde solche Umstellungen nie vornehmen. Zwei Personen waren sich nicht sicher, ob sie solche Umstellungen vornehmen, vermuteten jedoch, dass sie es über Kombinationen aus Markieren, Ausschneiden und Einfügen getan hätten. Zwei Personen hatten eine solche Funktion schon länger vermisst und gaben an, eine solche Umstellung unter Verwendung der Funktionen `kill-word`, `backward-kill-word`, `forward-word`, `backward-word` und `yank` vorzunehmen. Eine Person gab dafür auch ein Beispiel an; die Folge besteht aus elf Schritten (siehe Listing 8.1), das sind acht mehr als die optimale Folge von Schritten, die wir in Abbildung 7.2 auf Seite 195 gezeigt haben.

Listing 8.1: Tauschen von Konjunkten in XEmacs*, der Cursor ist vor dem ersten Konjunkt platziert.*

```
<S-C-right>          ;; forward-word mit Markierung der Zeichen
C-w                  ;; kill-region
<C-right>            ;; forward-word
C-y                  ;; yank
<S-C-right>          ;; forward-word mit Markierung der Zeichen
C-w                  ;; kill-region
3*<C-left>           ;; backward-word
<C-right>            ;; forward-word
C-y                  ;; yank
```

Die Rückmeldungen sind also insgesamt sehr positiv, Anregungen zur Veränderung wurden aufgenommen und umgesetzt. Die Antworten der verschiedenen Benutzer bestätigen die Aussagen, die Hausdorf [2005] aus der Evaluation des Werkzeugs *ScientiFix* gewinnt: Autoren sind bereit, neue Funktionen zu erlernen, besonders dann, wenn sie Möglichkeiten bieten, die ohne diese Funktionen nur mit Mühe erreichbar sind.

8.4 Ausblick

Für Untersuchungen zum konkreten Umgang mit verschiedenen Funktionen, insbesondere auch zur Frage, wie Informations-, Bewegungs- und Modifikationsfunktionen kombiniert werden, die den gleichen Skopus haben, müssen ausführliche Experimente durchgeführt werden. Zudem sollten zusätzlich Experimente zur Erreichung der gleichen Ziele ohne die von uns implementierten Funktionen durchgeführt werden. Dies hätte jedoch den Rahmen dieser Arbeit gesprengt.

Experimente zur Bedienfreundlichkeit von Editierfunktionen sollten in ein allgemeines Konzept zur Evaluation von Schreibwerkzeugen oder Schreibunterstützungssystemen eingebettet sein. Solche Konzepte existieren jedoch noch

nicht. Überspitzt lässt sich formulieren, dass mehr als zehn Jahre später die von Dale [1997] gestellten Fragen immer noch nicht gelöst sind. Es ist weiterhin offen, welche linguistischen Theorien sich für die Analyse und Korrektur von Syntaxfehlern eignen – schon 1989 fordert Cookson: «Writing is a linguistically based process, so designing automated tools should go hand in hand with a linguistic model.» [Cookson 1989, S. 20] – und ob man Syntax, Semantik und Pragmatik in Bezug auf das Erstellen natürlichsprachlicher Texte überhaupt voneinander getrennt betrachten kann. Es gibt kaum wirklich robuste Parser; für den Umgang mit unbekannten Wörtern oder falschgeschriebenen Wörtern gibt es noch keine befriedigenden Lösungen. Es gibt keine Konzepte zur Evaluation von Funktionen oder Programmen zur Schreibunterstützung: Sollte man auf Fehlerkorpora setzen, auf Daten zur Performanz oder zur produzierten Textmenge, auf die Zahl oder den Typ der (verbliebenen) Rechtschreib- und Syntaxfehler oder auf ihr Verhältnis? Wo sind Software-Theorien zur Unterstützung der kreativen Seite des Schreibens oder zur Einhaltung von stilistischen Vorgaben und Regeln? Erreichen implementierte Prototypen so selten den Markt, weil Vorstellungen und Terminologie von Entwicklern und Benutzern so weit auseinander liegen, dass man einander schlicht «nicht versteht» [siehe Dale 1997, S. 236f]?

Die vorliegende Arbeit liefert einen Anstoss, all diese Fragen anzugehen. Die technischen Möglichkeiten sind heute vielversprechend, das Interesse an der Beschäftigung mit diesen Fragen scheint in den letzten Jahren wieder zuzunehmen, wie die erfolgreichen Workshops im Rahmen von LREC 2008, SLTC 2008 und NAACL 2010 zeigen.

A

Sammlung von Redigierfehlern

Die folgende Übersicht enthält fehlerhafte Sätze; die entsprechenden Fehler sind auf misslungene Redigieroperationen zurückzuführen. Die Funde stammen aus Zeitungen («Süddeutsche Zeitung»[1], «Tages-Anzeiger»[2], «Neue Zürcher Zeitung»[3], «Südkurier»[4], «Uckermark Kurier»[5], «Der Tagesspiegel. Zeitung für Berlin und Deutschland»[6], «20 Minuten»[7], «Blick am Abend»[8], «Ostsee-Zeitung»[9], «Urlaubslotse»[10]), ihren Beilagen, Foren, Blogs («XING»), studentischen Arbeiten, Büchern sowie privat verfassten Texten. Fehler aus privat verfassten Texten wurden von den jeweiligen Autoren selbst übermittelt inklusive einer Erklärung (soweit aus der Erinnerung möglich), wie es zu diesem Fehler kam.

1. «Süddeutsche Zeitung. Nachrichten aus Politik, Kultur, Wirtschaft und Sport», Herausgeber: Süddeutsche Zeitung GmbH, Hultschiner Straße 8, 81677 München, Tageszeitung.
2. «Tages-Anzeiger», Herausgeber: Tamedia AG Zürich, Werdstrasse 21, 8021 Zürich, werktäglich.
3. «Neue Zürcher Zeitung, NZZ – Zeitung für die Schweiz», Herausgeber: Neue Zürcher Zeitung AG, Falkenstrasse 11, 8021 Zürich. werktäglich
4. «Südkurier. Unabhängige Tageszeitung in Baden-Würtemberg», Herausgeber: Gesellschafter der Südkurier GmbH, Max-Stromeyer-Str. 178, 78467 Konstanz, täglich ausser sonntags.
5. «Uckermark Kurier», Herausgeber: Kurierverlags GmbH & Co. KG, Friedrich-Engels-Ring 29, 17034 Neubrandenburg, werktäglich.
6. «Der Tagesspiegel. Zeitung für Berlin und Deutschland», Verleger: Dieter von Holtzbrinck, Herausgeber: Dr. Pierre Gerckens, Giovanni di Lorenzo, Dr. Hermann Rudolph, Verlag Der Tagesspiegel GmbH, Askanischer Platz 3, 10963 Berlin, täglich.
7. «20 Minuten», Herausgeber: Tamedia AG Zürich, Werdstrasse 21, 8021 Zürich, werktäglich.
8. «Blick am Abend», Herausgeber: Ringier AG, Brühlstrasse 5, 4800 Zofingen. Erscheint Montag bis Freitag.
9. «Ostsee-Zeitung. Die Unabhängige für Mecklenburg-Vorpommern. Ribnitz-Damgartner Zeitung», Herausgeber: OSTSEE-ZEITUNG GmbH & Co. KG, Richard-Wagner-Straße 1a, 18055 Rostock, Tageszeitung.
10. «Urlaubslotse. Die Zeitung für Feriengäste Fischland-Darss-Zingst», Herausgeber: OSTSEE-ZEITUNG GmbH & Co. KG, Lange Strasse 43/45, 18311 Ribnitz-Damgarten, erscheint von Anfang Mai bis Ende September wöchentlich freitags.

Einige Fehler, insbesondere die Kongruenz betreffend, könnten auch als Tippfehler erklärt werden. Die Autoren erinnern sich jedoch, dass diese Fehler durch Redigieroperationen verursacht sind. Nur wenige der Fehler können durch eine Grammatik- und Rechtschreibkorrektur erkannt werden, wie Gubelmann [2010] untersucht hat. In den Zeitungen nimmt die Zahl der Fehler in den letzten Monaten zu. Wenn man berücksichtigt, dass die Redaktionen nicht Korrekturprogramme abgeschafft haben, sondern Korrekturprogramme nie in den Redaktionssystemen vorhanden waren, lässt sich die zunehmend schlechtere Qualität der Texte nur mit der Personalpolitik der Redaktionen erklären, die letztlich auf die allgemeine finanzielle Lage der klassischen Zeitungen zurückzuführen ist: Korrektoren und Lektoren werden entlassen, die verbliebenen Redaktoren stehen unter einem enormen Zeitdruck.

Für jeden Fehler ist die Fundstelle angegeben. Das Feld *Kategorie* dient der Einordnung des Fehlers, unter *Kommentar* werden Erklärungen der jeweiligen Autoren (soweit bekannt) widergegeben, Vermutungen zu den Ursachen geäussert sowie Vorschläge gemacht, wie dieser Fehler hätte vermieden werden können. Die Kategorisierung der Fehler erfolgt auf verschiedenen Ebenen: Fehler können kategorisiert werden nach der betroffenen Wortart (z. B. Verben, Pronomen, Präpositionen), nach grammatikalischen Phänomenen (z. B. Infinitiv mit «*zu*»), nach allgemeineren strukturellen Eigenschaften (z. B. zu viele oder fehlende Wörter). Wir unterscheiden *doppelte Wörter* und *überflüssige Wörter*: In beiden Fällen enthält der Satz mehr Wörter als notwendig, um syntaktisch korrekt zu sein. Die Kategorie *doppelte Wörter* ist eine spezifische Variante, das überflüssige Wort (bzw. die überflüssigen Wörter) kommen schon einmal im Satz vor. Die meisten Fehler gehören verschiedenen Kategorien an.

Fehlertext		Quelle	Kategorie(n)	Kommentar
(A.1)	Hoeneß unterstrich zudem erneut den Willen des FC Bayern, den an Leverkusen Toni Kroos zurückzuholen.	Süddeutsche Zeitung, 28. Januar 2010, S. 37, «Hoeness contra Blatter» (sid)	Fehlende Wörter, Verben, Partizipien	Hier fehlt ein Verb wie «*ausgeliehenen*».
(A.2)	Gürtlers Dossier wurde auf einer der beiden Festplatten, die von der Bonner Staatsanwaltschaft sicher gestellt wurden.	Süddeutsche Zeitung, 5. November 2009, S. 15 «Das Raubtier aus Bonn» (Hans Leyendecker, Klaus Ott)	Fehlende Wörter, Verben	Hier fehlt das Vollverb für den ersten Teilsatz, etwa «*gefunden*».
(A.3)	Nach dem Blitzbesuch im Musikgeschäft fuhr Clintion [sic!] in seiner mit einer Eskorte davon.	Südkurier, 5. November 2009, «Blitzeinkauf von Bill Clinton in Zürich»[11]	Fehlende Wörter, Substantive, Präpositionalgruppen	Neben dem Tippfehler im Namen «*Clinton*» ist ein Fehler in der folgenden Präpositionalgruppe vorhanden. Entweder fehlt für «*in seiner*» das entsprechende Substantiv (etwa «*Limusine*»), oder diese Wortformen sind überflüssig und es sollte nur die Ergänzung «*mit einer Eskorte*» festgehalten werden.
(A.4)	Die Polizei hatte Alexejewa am Donnerstag etwa 50 anderen Oppositionellen festgenommen, die sich trotz Demonstrationsverbot im Zentrum der Russischen Hauptstadt versammelt hatten.	Süddeutsche Zeitung, 2./3. Januar 2010, S. 8, «Freilassung gefordert»	Fehlende Wörter, Präpositionen	Hier fehlt die Präposition «*mit*» für die Angabe «*mit etwa 50 anderen Oppositionellen*».
(A.5)	Margaret Atwood, 69, Schriftstellerin, verursachen Interviews mit Alpträume.	Süddeutsche Zeitung, 6. November 2009, S. 10, «Leute»	Fehlende Wörter, Substantivgruppe	Hier fehlt das Substantiv für die Präposition «*mit*».

11. Online-Ausgabe http://www.suedkurier.de/region/hochrhein/kanton-zuerich/Blitzeinkauf-von-Bill-Clinton-in-Zuerich;art372447,4019325 (zuletzt besucht am 10.11.2009, 19:30)

Fehlertext		Quelle	Kategorie(n)	Kommentar
(A.6)	Sie liegen 243 Punkte knapp vor Leif Lampater/Christian Grasmann (De/201 Punkte).	News, 16. November 2009, S. 40, «Newsticker Rad»	Fehlende Wörter, Präpositionen	Es müsste entweder heissen «*Sie liegen mit 243 Punkten knapp vor…*» oder «*Sie liegen knapp vor …*» Es geht um Radrennen.
(A.7)	Unsere Dezember-Ausgabe dieses Happenings wird mit einer Performance von Christoph Schlingensief am Freitagabend, 4. Dezember.	Das Magazin, Nr. 48 (28. November bis 4. Dezember 2009), S. 5, «Editorial» (Finn Canonica)	Fehlende Wörter, Verben	Hier fehlt das Vollverb des Satzes.
(A.8)	Pikant ist, dass diese Überbelastung letztlich nicht dem Unterricht zugutekommt, sondern – wie LCH-Zentralsekretärin Franziska Peterhans kritisch anmerkte.	Neue Zürcher Zeitung, 9. Dezember 2009, S. 12 «Reformdruck führt zu Fronarbeit» (Walter Hagenbüchle)	Fehlende Wörter, Konstruktion	In der Onlineausgabe[12] lautet der Satz: «*Pikant ist, dass diese Überbelastung letztlich nicht dem Unterricht zugutekommt, sondern – wie LCH-Zentralsekretärin Franziska Peterhans kritisch anmerkte – vor allem durch Gemeinschaftsarbeit im Schulteam, bei der Administration oder der Weiterbildung anfällt.*» Vermutlich musste für die Druckausgabe gekürzt werden.
(A.9)	Valdez versuchte, den Golden Retriever zu retten und erlitt dabei eine Rauchvergiftung ins Krankenhaus gebracht.	Süddeutsche Zeitung, 23. Dezember 2009, S. 10, «Leute»	Fehlende Wörter, Konstruktion	Die kurze Meldung geht bis zum letzten Zeichen auf der letzten Zeile, vermutlich musste extrem gekürzt werden. Mögliche korrekte Varianten sind «*…Rauchvergiftung, mit der er ins Krankenhaus gebracht wurde*» oder «*…Rauchvergiftung und wurde ins Krankenhaus gebracht.*»
(A.10)	Als seine Wut an einem Ameisenhügel auslässt, bekommt er arge Probleme.	Uckermark Kurier, 24./25. Dezember 2009, S. 11, «Lucas, der Ameisenschreck», Bildunterschrift zur Ankündigung des Films	Fehlende Wörter, Pronomen	Die beiden Zeilen sind bis zum Ende gefüllt, es ist kein Platz mehr für «*er*».

12. http://www.nzz.ch/services/lehrer_verband_pflichtstunden_senkung_forderung_1.4132972.html?printview=true (zuletzt besucht am 8.12.2010, 19:29)

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.11) Dazu brauchte die beiden Pioniere aus Wengen und ihren verletzten Stolz.	Tages-Anzeiger, 8. Januar 2010, S. 51, «Der verletzte Stolz der Pioniere» (Martin Born)	Fehlende Wörter, Pronomen	Man kann nicht erraten, wie der Satz richtig heissen könnte. Berücksichtigt man den direkt vorangehenden Satz, kann man ihn immerhin verstehen: «*Allein hätte die Berner Oberländer Dreifaltigkeit aus Eiger, Mönch und Jungfrau (und schon gar nicht der Wildstrubel in Adelboden) so etwas nie geschafft.*» Also fehlt im obigen Satz ein «*es*»: «*Dazu brauchte es die beiden …*»
(A.12) Insgesamt haben die Archäologen in dem Waldstück in der Nähe des Kalefelder Ortsteils Oldenrode rund 1000 Fundstücke römischer Herkunft aus dem Boden geholt, Waffen und Ausrüstungsgegenstände von Legionären, aber auch um Silbermünzen.	Uckermark Kurier, 6. Januar 2010, S. 21, «Neue Grabung auf antikem Schlachtfeld»	Fehlende Wörter, Verben	Für den letzten Teilsatz fehlt das Verb. Funktionieren würde: «*Insgesamt haben die Archäologen …aus dem Boden geholt, Waffen und Ausrüstungsgegenstände von Legionären. Es handelt sich aber auch um Silbermünzen.*» Der zweite Satz klingt nicht so schön. Vielleicht war es also ursprünglich: «*Insgesamt haben die Archäologen …aus dem Boden geholt. Es handelt sich um Waffen und Ausrüstungsgegenstände von Legionären, aber auch um Silbermünzen.*»
(A.13) Die Einreihung richtet sich im Sinne der allgemeinen Bestimmungen nach der Lohnklasse gemäss Stellenplan oder wird aufgrund einer Arbeitsbewertung gemäss Vereinfachter Funktionsanalyse (VFA, vgl. auch Unterlage III.1.4).	Lohntabelle für LR 01 und 05; gültig ab 1. Januar 2008 für den Kanton Zürich (Kantonale Verwaltung, Handbuch Personalrecht), Absatz «2.2 Einreihungsregeln»	Fehlende Wörter, Verben	Für den zweiten Teilsatz fehlt das Verb. Eine Erklärung ist, dass beim Einfügen der erläuternden Klammer versehentlich das Verb (etwa «*festgelegt*») markiert und überschrieben wurde. Eine andere Erklärung ist, dass während des Schreibens dieses Satzes entschieden wurde, eine Erläuterung für «*Vereinfachte Funktionsanalyse*» anzugeben, und der eigentliche Satz noch nicht vollständig war.
(A.14) Dabei es zum vielfältige Arbeitsbereiche	Online Beitrag der UZH News, 3. April 2009, «Multilinguale Textanalyse. Vieldeutigkeit der Sprachen verstehen» (Marita Fuchs)[13]	Fehlende Wörter, Verben	Hier wurde mehrfach editiert. Das Überbleibsel «*zum*» kann aus einer Konstruktion wie «*zum Beispiel*» herrühren. Während des Umstellens ging das Verb verloren oder wurde nicht eingefügt.

13. http://www.uzh.ch/news/articles/2009/vieldeutigkeit-der-sprachen-verstehen.html (zuletzt besucht am 5.4.2009, 19:29), mittlerweile wurde der Fehler behoben.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.15) Nach sechsjähriger guter Zusammenarbeit ist Oliver mein engster Mitarbeiter und Vertrauter, deshalb wird er nach der WM auch der Erste sein, mit ich am Tisch sitze, um zu beraten, wie es für uns weitergeht und was wir wollen.	Süddeutsche Zeitung, 10. Februar 2010, S. 33, «Was sind wir doch für Hornochsen» (Marc Widmann, Thomas Kistner)	Fehlende Wörter, Artikel	Der Satz ist ein wörtliches Zitat, wobei der Sprecher den Artikel «*mit [dem] ich*» sicher nicht vergessen hat, er ist erst beim Abschreiben oder Redigieren verschwunden.
(A.16) Sie sich viel Zeit für den Bettenkauf.	NZZ am Sonntag, 14. Februar 2010, S. 83, «5 Tipps zum Kauf» Infokasten zum Beitrag «Für den besseren Schlaf»	Fehlende Wörter, Verben	Eventuell sollten die Empfehlungen so formuliert werden, dass das Verb für alle aufgelisteten Regeln in der Einleitung enthalten ist. Alle anderen Tipps sind jedoch als vollständige Sätze formuliert.
(A.17) Andere von ihm angesprochene Aktivitäten, beispielsweise in Arbeitsgruppen des europäischen Fußballverbandes Uefa, berühren keineswegs die Ämter von deutschen Funktionsträgern dort, vielmehr rangieren auf nachgeordneter Ebene.	Süddeutsche Zeitung, 10. Februar 2010, S. 33, «Was sind wir doch für Hornochsen» (Marc Widmann, Thomas Kistner)	Fehlende Wörter, Pronomen	Hier fehlt im letzten Teilsatz das Pronomen «*sie*». In einer anderen Wortstellung würde das Pronomen nicht unbedingt notwendig sein: «*…, berühren keineswegs die Ämter …dort, rangieren vielmehr auf …*».
(A.18) Obszönität aber ist ein Verhältnisausdruck, hinter sich jetzt schon seit Jahrzehnten eine antiphilosophische Mytik umtreibt: der Glaube an «das Leben».	Süddeutsche Zeitung, 11. Februar 2010, S. 15, «Ich bin in Berlin. Es geht um meinen Wahn» (Thomas Steinfeld)	Fehlende Wörter, Pronomen	Hier fehlt das Pronomen «*dem*» im Relativsatz.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.19) Für ihre zweite Regiearbeit nach «Filth and Wisdom» habe sich Madonna die Liebesgeschichte zwischen Edward VIII. und der geschiedenen Gesellschaftsdame Wallis Simpson.	Süddeutsche Zeitung, 16. Februar 2010, S. 38, «Karnevalisten»	Fehlende Wörter, Verben	Hier fehlt das Hauptverb des Satzes.
(A.20) So wurden ab 2000 viele Einzelversuche und Pilotprojekte mit Portfolio-Arbeit in Schule, in Hochschule (v.a. in der Lehrerbildung) und in der beruflichen Bildung.	«Konzept ePortfolio des Master-Studiums der Hochschule für Soziale Arbeit FHNW», S. 1, Stand August 2009 (Esther Forrer Kasteel, Elisabeth Müller Fritschi)	Fehlende Wörter, Verben	Hier fehlt ebenfalls das Hauptverb des Satzes.
(A.21) Dennoch sind die Übergänge heute meist fliessend und eine Abgrenzung zwischen den beiden Komponenten häufig nur schwer zu definieren.	Seminararbeit von Reto Seeholzer, Oktober 2009	Fehlende Wörter, Verben, Kongruenz	Hier sind zwei Hauptsätze durch «*und*» miteinander verbunden, dem zweiten Satz fehlt das finite Verb. Teilen sich beide Sätze das gleiche finite Verb, könnte es weggelassen werden, dies ist hier jedoch nicht möglich.
(A.22) Schon zu aktiven Zeiten galt er als extrem autoritärer Spielleiter, als führender Funktionär hat den strengen Stil nicht abgelegt.	Süddeutsche Zeitung, 10. März 2010, S. 28, «Der ausgefallene Rücktritt» (Philipp Selldorf)	Fehlende Wörter, Pronomen	Hier fehlt das «*er*», dann wäre der Teilsatz vollständig: «*als führender Funktionär hat er den strengen Stil nicht abgelegt*».
(A.23) Noriega hatte sich seit 2007 erbittert die Auslieferung gewehrt.	Der Tagesspiegel, 28. April 2010, S. 5, «Noriega an Frankreich ausgeliefert» (DPA)	Fehlende Wörter, Präpositionen	Hier fehlt die Präposition «*gegen*» vor «*die Auslieferung*».

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.24) Die Römer hatten es im Laufe ihrer Geschichte mehrmals gegnerischen Kriegselefanten zu tun.	Süddeutsche Zeitung, 26. März 2010, S. 18 «Zehn Dinge die Sie noch nicht wissen über Schmerz» (Sebastian Herrmann)	Fehlende Wörter, Präpositionen	Hier fehlt das «*mit*» vor «*gegnerischen Kriegselefangen*».
(A.25) Und sein Berater Jean-Prierre Bernès ist zwar beim Spiel gewesen, aber Informationen verweigerten die Bayern dem windigen Geschäftsmann aus Cassis eine Audienz.	Süddeutsche Zeitung, 23. April 2010, S. 31, «Menschliche Schwächen» (Andreas Burkert)	Fehlende Wörter	Vermutlich fehlen vor «*Informationen*» die Wörter «*nach unseren*», dann würde der Satz syntaktisch korrekt sein.
(A.26) Die Strassenopfer-Stiftung Roadcross fährt Rasern mit Vollgas an den Karren. Sie will, dass die Verfassung definiert, wer als «Raser» gilt und wie hart die ausfallen.	Blick am Abend (Ausgabe Zürich), 27. April, 2010, S. 2, «Das Volk gegen die Raser» (Ann Guenter)	Fehlende Wörter	Zitiert ist hier der komplette Anfang des Beitrags. Aus dem Kontext erschliesst sich, dass es im letzten Teilsatz um Strafen geht. Jedoch müsste dies explizit gemacht werden, damit der zweite Satz akzeptabel wäre – «*die*» wäre dann als Anapher zu lesen. Oder das «*die*» ist tatsächlich ein Artikel und «*Strafen*» fehlt.
(A.27) Für praktische Fahrstunden müssten die Schüler oder deren Eltern jeweils 24 bis 32 Euro berappen. Am Ende lägen die Kosten bei deutlich über 1000 Euro. Wiederholer müssen nach meist noch einen Batzen drauflegen.	Uckermark Kurier, 6. April 2010, S. 1, «Fast die Hälfte der Fahrschüler fällt durch» (DPA)	Fehlende Wörter	Im letzen Satze fehlt ein Objekt für die Präposition «*nach*». Entweder fehlt hier etwas Ähnliches wie «*nicht bestandener Prüfung*» oder die Angabe einer Person, die diese Aussage machte.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.28) Aber diese wirklich unbedeutende Niederlage werden die Bayern gerne in Kauf, schließlich ist ihr spielendes Personal erkennbar auf der Höhe der Zeit.	Süddeutsche Zeitung, 3. Mai 2010, S. 25, «Der Meister und das Biest» (Andreas Burkert)	Fehlende Wörter, Verben	Hier fehlt nach «*in Kauf*» das Verb «*nehmen*». Unter Umständen hiess eine erste Fassung: «*…Niederlage nehmen die Bayern gerne in Kauf, schließlich …*».
(A.29) Dem lag der Gedanke zugrunde, dass es schnell unglaubwürdig und anmaßend werden kann, wenn hochbezahlte Stadt- oder Staatstheaterschauspieler versuchen, naturalistisch den Underdog aus China spielen.	Der Tagesspiegel, 19. Mai 2010, S. 23, «Das innere Schauen» (Bernahrd Schulz)	Fehlende Wörter, Infitinitiv mit «*zu*»	Vermutlich wurde «*versuchen*» erst später eingefügt und dann übersehen, dass der letzte Teilsatz nun ein erweiterter Infinitiv mit «*zu*» sein muss.
(A.30) Durch die Zusammenarbeit mit Archäologen sind Ethnologen heute aber auch in der Lage, Kontinuitäten in der Kultur der Maya feststellen.	Süddeutsche Zeitung, 10./11. Oktober 2009, S. 22, «Die Maya-Renaissance» (Sophie & Marcel Burkhardt)	Fehlende Wörter, Infinitiv mit «*zu*»	Wird statt der Formulierung «*sind …in der Lage*» lediglich «*können*» verwendet, wird kein «*zu*» benötigt.
(A.31) Man vermag sich den Kibbuznik Chomsky lebhaft vorstellen: Ein freier Geist, beseelt vom Glauben an eine bessere Welt, überzeugt vom Gebot der Gerechtigkeit.	Neue Zürcher Zeitung, 20. Mai 2010, S. 2, «Eine verhinderte Reise nach Palästina» (Martin Woker)	Fehlende Wörter, Infinitiv mit «*zu*»	Das Modalitätsverb «*vermögen*» verlangt den Infintiv mit «*zu*», das Modalverb «*können*», das evtl. in einer früheren Version dieses Satzes verwendet wurde, verlangt dies nicht.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.32) Vor anderthalb Jahre starb Sohn des Ehepaars im Alter von 16 Jahren während eines Familienurlaubs.	Süddeutsche Zeitung, 20. Mai 2010, S. 10, «Leute»	Fehlende Wörter, Artikel, Substantivgruppen	Hier fehlt entweder ein Artikel («*ein*» oder «*der*») oder sogar Artikel und Adjektiv («*der älteste*», o. ä.).
(A.33) Fragebogen für Angehörige von Suchtkranken 17. Ich sage meinem Partner, er sich mich verlassen, bis er mit dem Konsum aufgehört hat, folge ihm danach aber doch sofort, um zu sehen, was er macht.	Prenzlauer Zeitung, 4. Juni 2010, S. 14, «‹Fangen Sie an, an sich selbst zu denken›. Alkoholmissbrauch» (Claudia Marsal)	Fehlende Wörter, Verben, Präpositionen	Hier sind mehrere korrekte Varianten möglich: «*…er soll mich verlassen, bis er mit dem …*» oder «*…er kann sich auf mich verlassen, bis er mit dem …*»
(A.34) Bei 25cm-Wassertiefe und 34 Grad Wassertemperatur haben junge Mütter und sogar Väter mit ihren Babys.	Urlaubslotse, 30. Juli bis 5. August 2010, S. 10, «Bei kaltem Wetter lockt die Bodden Therme!» (F.B.)	Fehlende Wörter	Hier fehlt nach «*Väter*» oder nach «*Babys*» die Angabe, was Mütter und Väter «*haben*», etwa «*Spass*».
(A.35) Das Kunstmuseum Ahrenshoop soll zu einem Ausstellungs-, Begegnungs- und Forschungszentrum des Künstlerortes, heißt es seitens des Vereins.	Ostsee-Zeitung, 4. August 2010, S. 13, «Museum: Ungeteilte Zustimmung ist dahin. Die Parkplatznot in Ahrenshoop bekommen die Initiatoren des Kunstmuseums zu spüren. Die pochen auf früher gemachte Verabredungen» (Timo Richter)	Verben	Hier fehlt das finite Verb für den ersten Teilsatz.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.36) Das Stipendienprogramm gilt als Prestigeobjekt der FDP in der Koalition und wurde gegen erhebliche Bedenken auch unionsgeführter durchgesetzt.	Uckermark Kurier, 28. Juli 2010, S. 2, «Bund speckt Stipendienprogramm ab» (DPA)	Fehlende Wörter, Substantive	Hier fehlt nach dem Adjektiv «*unionsgeführter*» das Substantiv.
(A.37) Von Freunden, denen man überhaupt erst explizit verbieten muss, ihre Aufnahmen der torkelnden Braut ins Netz zu stellen, kann auch nach der Hochzeit wenig Gutes erwarten.	Süddeutsche Zeitung Magazin, Nr. 31, 6. August 2010, S. 23, «Eine Frage des (Ver)Trauens. Neuerdings verbieten Hochzeitspaare ihren Gästen, Bilder und Videos vom Fest ins Internet zu stellen. Das verspricht nichts Gutes für die Ehe» (Till Krause)	Fehlende Wörter, Pronomen	Hier müsste das «*man*» im letzten Teilsatz noch einmal wiederholt werden.
(A.38) Das Interesse Kunstwerken aus der legendären Ahrenshooper Künstlerkolonie ist gewaltig.	Uckermark Kurier, 4. August 2010, S. 21, «Bilder aus Ahrenshoop zur Versteigerung» (Anette Pröber)	Fehlende Wörter, Präpositionen	Korrekt wäre etwa «*Das Interesse an Kunstwerken …*»
(A.39) Der Bundestrainer hat der abenteuerlich anmutenden Absprache der beiden Verbände zugestimmt, hieß es DFB.	Süddeutsche Zeitung, 27. August 2010, S. 27, «Verlängerung bis zur Geisterstunde. Bundestrainer Löw plant offenbar, Michael Ballack wegen des Formrückstands noch nicht für die ersten Qualifikationsspiele zu berufen» (Philipp Selldorf)	Fehlende Wörter, Präpositionen, Artikel	Der Nebensatz könnte korrekt heissen: «*hieß es beim DFB*» oder «*hieß es vom DFB*».

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.40) In einem am Donnerstag erschienenen Buch greift der frühere Team-Arzt, Jean-Pierre Paclet, einen schon kurz nach dem Titelgewinn vor zwölf Jahren publik gewordenen Doping-Verdacht gegen wieder auf.	Süddeutsche Zeitung, 27. August 2010, S. 27, «Franzosen am Pranger. Vorwürfe gegen WM-Team von 1998» (DPA)	Fehlende Wörter	Hier fehlt der Name desjenigen, gegen den die Vorwürfe erhoben werden.
(A.41) Da waren Fragen aufgetaucht, als bekannt wurde, dass der Drittligist SV Babelsberg, dem der damalige Finanz- und heutige Innenminister Rainer Speer (SPD) Millionenbeträge für die Sanierung seines Stadions erhalten wird.	Uckermark Kurier, 30. August 2010, S. 4, «Ein Ministerfreund mit zweifelhafter Vergangenheit» (Johann Legner)	Fehlende Wörter	Es fehlt etwas, um die Rolle des Innenministers zu spezifizieren und den mit «*dem der*» eingeleiteten Nebensatz zu beenden, der eingebettet ist in «*dass der Drittligist SV Babelsberg Millionenbeträge für die Sanierung seines Stadions erhalten wird.*»
(A.42) Wie jeder weiß, werfen wir uns nicht Wörter um die Ohren, sondern meistens.	Thomas Hanneforth: «Computerlinguistik. Computer lernen unsere Sprache», online [14]	Fehlende Wörter	Hier wird der Satz nicht beendet.
(A.43) Ausserdem muss der Klub im nächsten Spieler-Draft auf zwei Ziehungen – auf eine in der ersten und eine in der dritten Runde.	Neue Zürcher Zeitung, 15. September 2010, S. 51, «New Jersey Devils massiv gebüsst. Folgen des Kowaltschuk-Deals» (ped)	Fehlende Wörter, Verben	Hier fehlt das finite Verb nach «*Ziehungen*», möglich wäre «*verzichten*».

14. http://www.ling.uni-potsdam.de/index.php/de/studium/bachelorstudiengaenge/bsc-computerlinguistik/was-ist-computerlinguistik.html (zuletzt besucht am 8.12.2010, 19:42).

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.44) Es erzählt das Liebesleben der weiblichen Hauptfigur von ersten Teenagersehnsüchten bis zur späten pragmatischen Beziehung zu einem jungen Afrikaner, schon in dem Alter ist, in dem andere Frauen Großmutter werden.	Uckermark Kurier, 11./12. September 2010, S. 31, «Variantenreiche Geschichten vom Konfliktpotenzial der Liebe. Kolumnist Harald Martenstein nimmt mit «Gefühlte Nähe» souverän die Hürde des zweiten Romans» (Andreas Heimann)	Fehlende Wörter, Pronomen	Hier fehlt das Relativpronomen, das den Bezug zur «*weiblichen Hauptfigur*» herstellt, es müsste heissen «*…Afrikaner, die schon in einem Alter ist …*».
(A.45) Demach wurde Pastior, der unter anderem wegen einiger antisowjetischer Gedichte, die er während seiner Gefangenschaft geschrieben hatte, am 8. Juni die Zusage zur Mitarbeit abgenötigt.	Süddeutsche Zeitung, 17. September 2010, S. 1, «Der verstrickte Gefährte. Herta Müllers Freund Oskar Pastior war der Securitate verpflichtet» (Christopher Schmidt, Lothar Müller)	Fehlende Wörter, Verben	Im eingeschobenen Nebensatz «*der unter anderem wegen einiger antisowjetischer Gedichte*» fehlt ein finites Verb.
(A.46) Karp, damals 33, wurde 1999 in Wildau.	Tagesspiegel, 8. Oktober 2010, S. 15, «90 000 Euro Schaden. CDU-Staatssekretär wurde Manager in Wolfsburg. Brandenburg kommt das teuer zu stehen» (Thorsten Metzner)	Fehlende Wörter, Verben	Aus dem Kontext geht hervor, dass das fehlende Verb vermutlich «*beurlauben*» ist.
(A.47) Die Terrorismus und Spionage zuständige Ermittlungsbehörde prüft, ob der Anfangsverdacht einer geheimdienstlichen Straftat vorliegt.	Uckermark Kurier, 22. März 2010, S. 5, «Abhörtechnik beschäftigt Landtag» (Alexander Fröhlich)	Fehlende Wörter, Präpositionen	Korrekt wäre «*Die für Terrorismus und Spionage zuständige Ermittlungsbehörde*». Vermutlich sollte «*Die zuständige Ermittlungsbehörde*» erweitert werden und es wurde eingefügt, wofür die Behörde zuständig ist.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.48) Diese «Verschiebung» der Arbeitslosigkeit könnte damit zu haben, dass sich jugendliche Arbeitskräft, die landesweit mit einer Arbeitslosenquote von über 20 % konfrontiert sind, gar nicht mehr um eine Stelle benühen bzw. sich mit einem Praktikum als Alternative zufriedengeben.	Neue Zürcher Zeitung, 28. Oktober 2010, S. 29, «Zu wenig Arbeit in Frankreich. Steigende Arbeitslosigkeit» (rt)	Fehlende Wörter, Verben	Korrekt wäre «*Diese «Verschiebung» der Arbeitslosigkeit könnte damit zu tun haben …*».
(A.49) Bis zum Schluss Aziz dem Diktator treu.	Neue Zürcher Zeitung, 27. Oktober 2010, S. 5, «Todesstrafe für Saddams Aussenminister Aziz. Ein Gerichtsurteil mit politischer Sprengkraft im Irak» (Inga Rogg)	Fehlende Wörter, Verben	Hier fehlt das Verb. Korrekt wäre etwa «*Bis zum Schluss blieb Aziz dem Diktator treu*.»
(A.50) Das Ostholstein-Museum in Eutin eine Ausstellung mit Aquarellen, Zeichnungen und Ölbildern des Künstlers. «Armin Mueller-Stahl. Malerei und Papierarbeiten» heißt die Schau, die am 5. Dezember eröffnet wird und bis zum 30. Januar zu sehen ist.	Uckermark Kurier, 26. November 2011, S. 21, «Jetzt singt er doch noch. Debüt. Armin Mueller-Stahl ist ein Ausnahme-Schauspieler» (Nada Weigelt, DPA)	Fehlende Wörter, Verben	Hier fehlt im ersten Satz das Verb. Korrekt wäre etwa «*Das Ostholstein-Museum in Eutin zeigt eine Ausstellung*».
(A.51) Bewerbungen: Ab Frühjahr 2011 können Sie sich wieder für den Christkindlimarkt.	Webseite des Christkindlimarkt Zürich 2010, http://www.christkindlimarkt.ch/service/ (zuletzt besucht am 7.12.2010, 13:45)	Fehlende Wörter, Verben	Hier fehlt ebenfalls das Verb.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.52) Zum anderen bereite ich ein Projekt vor, **in dem es um** den Einsatz von Web 2.0 **in dem es um** die Weiterbildung älterer Arbeitnehmer geht.	Headz Blog, 1. November 2009, «Silver Surfer – Senioren im Internet» (Matthias Rohs)[15]	Doppeltes Auftreten von Wörtern, Einleitung von Nebensätzen	Hier wurden vermutlich Informationen aus zwei Sätzen zusammengefügt, die gleich aufgebaut waren.
(A.53) Auch die meisten Winterzugänge tragen das Merkmal der Jugend, aber da es jetzt ernst wird, **wird es** für Matip & Co **wird es** nun schwer werden, ihr angewandtes Praktikum in der Liga fortzusetzen.	Süddeutsche Zeitung, 15. Januar 2010, S. 11, «Trau keinem über 30» (Moritz Kielbassa)	Doppeltes Auftreten von Wörtern	Funktionieren würde: «*Auch die meisten Winterzugänge tragen das Merkmal der Jugend, aber für Matip & Co wird es nun schwer werden …*» Vielleicht wurde die Ernsthaftigkeit also noch eingeschoben und dann musste der Anschluss umformuliert werden und das ursprüngliche Verb blieb übrig.
(A.54) Die Büste repräsentiere **Ägypten** in Deutschland **Ägypten** und sei damit für die Kultur und Geschichte ihrer Heimat der beste Werbeträger.	Süddeutsche Zeitung, 4. Januar 2010, S. 31, «Die besten Blogs über ‹Nofretete›» (Karin Minawi)	Doppeltes Auftreten von Wörtern, Namen	Die Stellung von «*Ägypten*» ist an beiden Orten möglich, hier wurde offenbar probiert und die nichtgewählte Variante nicht entfernt.
(A.55) Dabei **ist** Fälschung in der Forschung so alt **ist** wie die Wissenschaft selbst.	Neue Zürcher Zeitung, 26. Oktober 2009, S. 9, «Wäre es nur so einfach wie bei einem Diebstahl»	Doppeltes Auftreten von Wörtern, Verben	Hier wurde eventuell mit Varianten der Einleitung experimentiert: «*dabei*» und «*wobei*» erzwingen eine andere Stellung des finiten Verbs.
(A.56) Wir **fliegen** wie geplant am 30. Oktober nach Kairo **fliegen**.	Süddeutsche Zeitung, 26. Oktober 2009, S. 16, «Marwa zu Ehren» (Karin El Minawi)	Doppeltes Auftreten von Wörtern, Verben	Hier wurde eventuell mit dem Futur experimentiert.

15. http://2headz.ch/blog/?p=1070 (zuletzt besucht am 8.12.2010, 19:26).

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.57) Ideal ist eine Zimmertemperatur von 20 Grad Celsius oder etwas höher. *Anfangs wird nur sehr wenig gegossen. Erst wenn die Blütenknospe erscheint und der Blütenstiel etwa handhoch ist, erhält die Amaryllis mehr Wasser. Übrigens: Werden die Zwiebeln etappenweise im Abtand von zwei bis drei Wochen gepflanzt, hat man die Möglichkeit, sich über einen längeren Zeitraum an blühenden Amaryllis zu erfreuen.* **Anfangs wird nur sehr wenig gegossen. Erst wenn die Blütenknospe erscheint und der Blütenstiel handhoch ist, erhält die Amaryllis mehr Wasser. Übrigens: Werden die Zwiebeln etappenweise im Abstand von zwei bis drei Wochen gepflanzt, hat man die Möglichkeit, sich über einen längeren Zeitraum an blühenden Amaryllis zu erfreuen.** Mit ein paar Tricks kann man die Amaryllis auch im nächsten Jahr …	Weihnachtsbeilage zum Uckermark Kurier, 24./25./26. Dezember 2009, S. 21, «Ein Star ohne Allüren» (nom/GP)	Doppeltes Auftreten von Wörtern	Hier wurden gleich mehrere Sätze wiederholt. Interessant ist, dass der Absatzumbruch jeweils unterschiedlich ist.
(A.58) In einem Steinbruch wurde nun eine männliche Leiche gefunden, nach Aussagen von Stolbunow **ist** es der Vermisste **ist**.	Süddeutsche Zeitung, 24. Juli 2009, S. 8, «Menschenrechtler tot aufgefunden» (zri)	Doppeltes Auftreten von Wörtern, Verben	Das doppelte finite Verb stammt wohl auch aus Experimenten mit dem Nebensatz, vielleicht hiess der zwischenzeitlich «*…, die nach Aussagen von Stolbunow der Vermisste ist.*»

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.59) Dabei **muss** mindestens eine Betreuungsperson bzw. ein Mitglied der Promotionskommission aus dem Kreis der am Doktoratsprogramm beteiligten Professoren stammen **muss**.	Entwurf «Wegleitung zum Doktoratsprogramm Linguistik an der Universität Zürich», S. 2, Stand November 2009	Doppeltes Auftreten von Wörtern, Verben	Eine Erklärung wäre, dass der Satz zwischendurch «*Wobei mindestens eine …stammen muss*» hiess. Dies war vielleicht auch ein Nebensatz zu dem vorhergehenden, der insgesamt gekürzt wurde, also wurde der Satzanfang auf «*Dabei muss mindestens eine …*» geändert – und das finite Verb aus der Ursprungsfassung hinten vergessen.
(A.60) Als internationaler Wochenaufenthalter **können** Sie mittels Tarifkorrekturantrag bis jeweils 31.03.des Folgejahres zusätzliche Abzüge **geltend machen können**.	XING-Forum «Deutsche in der Schweiz», 13. Dezember 2009 10:57, «Re: Quellensteuer Zürich, welcher Tarif und welcher Satz» (Dr. Peter Lutz)[16]	Doppeltes Auftreten von Wörtern, Verben	Hier wurden eventuell Versatzstücke aus verschiedenen Quellen zusammenkopiert.
(A.61) Auch **hat** die Evaluation gezeigt **hat**, dass Merkmale bei der Auflösung von verschiedenen Arten von Anaphora unterschiedliche Gewichtungen erhalten.	Don Tuggener, «Machine Learning für Koreferenz-Auflösung», Lizenziatsarbeit der Philosophischen Fakultät der Universität Zürich, Oktober 2009, S. 92	Doppeltes Auftreten von Wörtern, Verben	Hier wurde vermutlich mit dem Satzanfang experimentiert, eventuell handelt es sich um die Auftrennung eines längeren Satzes.
(A.62) Über 20 Jahre Entwicklung **hat** die CAT-Technologie **hat** dazu geführt, dass der technische Stand sehr hoch ist.	Studentische Arbeit, Herbstsemester 2009 (berichtet von Michael Piotrowski)	Doppeltes Auftreten von Wörtern, Verben	Vermutlich ist dieser Fehler durch nachträgliches Einfügen von «*CAT-Technologie*» entstanden oder durch die nachträgliche Erweiterung des ursprünglichen Satzes um den jetzigen Anfang («*Über 20 Jahre Entwicklung*»).

16. https://www.xing.com/net/germanyswitzerland/krankenkasse-versicherungen-steuern-finanzanlagen-private-vorsorge-35814/quellensteuer-zurich-welcher-tarif-und-wel 26629177/#26629177 (zuletzt besucht am 8.12.2010, 19:26)

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.63) Zusätzlich ist zu berücksichtigen, dass das Werkzeug **hat** natürlich auch Einfluss auf den Prozess **hat**.	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Doppeltes Auftreten von Wörtern, Verben	Hier wurde der Satz «*Das Werkzeug hat natürlich auch Einfluss auf den Prozess.*» in einen Nebensatz umgewandelt. Das finite Verb wurde korrekt an das Ende des Nebensatzes gesetzt, es ist jedoch auch an der ursprünglichen Stelle stehengeblieben.
(A.64) Dessen Rennstallchef, der frühere Schweizer Profi Mauro Gianetti, **hatte** 1998 mal einige Tage im Koma gelegen **hatte**, nachdem er bei der Tour de Romandie kollabiert war.	Süddeutsche Zeitung, 19./20. Juli 2008, S. 35, «Remscheid prüft. Alle Indizien deuten auf systematisches Team-Doping bei Saunier Duval hin – alimentiert von einem deutschen Unternehmen» (Andreas Burkert, Thomas Kistner)	Doppeltes Auftreten von Wörtern, Verben	Wahrscheinlich wurde hier nachträglich ein Einschub vorgenommen oder verschiedene Sätze zusammengefügt. Korrekt sind die Varianten «*Dessen Rennstallchef hatte 1998 mal einige Tage im Koma gelegen, nachdem er …*» oder «*…, der frühere Schweizer Profi Mauro Giannetti, der 1998 mal einige Tage im Koma gelegen hatte, …*».
(A.65) David packt die Fotos von seinem Unfallort aus. Auf einem **ist** sein Finger **ist** zu sehen.	Süddeutsche Zeitung, 12. Februar 2010, S. 10, «Auf Recherche in Freyung» (Max Hägler)	Doppeltes Auftreten von Wörtern, Verben	Eine frühere Version könnte «*David packt die Fotos von seinem Unfall aus, auf denen sein Finger zu sehen ist.*» gewesen sein.
(A.66) Es **ist** wie ein Organismus **ist**, der sich von seinem eigenen Körper ernährt.	Süddeutsche Zeitung, 23./24. Januar 2010, «‹Google ist das Äquivalent zur Kommunistischen Partei› Jaron Lanier warnt vor dem Kult des Kollektivs im Internet – die heutige Online-Kultur hält er für Digitalen Maoismus» (Interview: Jörg Häntzschel)	Doppeltes Auftreten von Wörtern, Verben	Auch hier liegt die Vermutung nahe, dass es sich ursprünglich um einen Nebensatz handelte.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.67) Pfeifend setzen sich die Rotoren der **vierzig 40** Jahre alten elektrischen Schneeschleuder in Bewegung.	Süddeutsche Zeitung, 21. Januar 2010, S. 29, «Spur gut!» (Sven Weniger)	Verdopplung	Der Autor hat wahrscheinlich nachträglich die Schreibung an den Hausstil angepasst. Im Originallayout ist der Zeilenumbruch genau zwischen «*vierzig*» und «*40*»; wenn der Autor die Änderung in dieser Ansicht vorgenommen hat, könnte das auch dazu beigetragen haben, dass ihm der Fehler nicht aufgefallen ist.
(A.68) Ja, ja, ja, sie habe sich bei Airens Blog bedient, gibt sie zu, warum auch nicht, der **habe** doch «einen Teil der alternativen Lebensweise, über die ich berichten wollte, auf den Punkt gebracht **hat**, und mit dem ich über das Buch auch ein Stück weit versuche, in Kommunikation zu treten».	Süddeutsche Zeitung, 9. Februar 2010, S. 11, «Untermieter im eigenen Kopf» (Willi Winkler)	Überflüssige Wörter, Verben	Durch das Zusammenfügen von indirekter und direkter Rede ist das finite Verb am Ende doppelt vorhanden.
(A.69) **Ihret**wegen **ihr** wurde über die olympische Bob- und Rodelbahn zuletzt viel geredet und geschrieben.	Süddeutsche Zeitung, 11. Februar 2010, S. 39, «Fifty-fifty durch die wilde 13» (Volker Kreisl)	Überflüssige Wörter, Pronomen	Im Umbruch ist «*Ihretwegen*» getrennt, so dass die folgende Zeile beginnt mit «*wegen ihr wurde über*», und das wäre grammatisch – lässt man das Ende der vorigen Zeile ausser acht. So wurde der Fehler vermutlich nicht bemerkt.
(A.70) Dabei soll die Wahrscheinlichkeit mit der das gewünschte Wort in der Liste der Wortvorschläge befindet möglichst gross sein, gleichzeitig soll **jedoch** sie **jedoch** der Übersicht halber möglichst wenige Wortvorschläge beinhalten.	Seminararbeit Reto Seeholzer, November 2009	Doppeltes Auftreten von Wörtern, fehlende Wörter, Konjunktion, Pronomen	Insgesamt ist der Satz schlecht lesbar, hier wurden möglichst viele Informationen in einen Satz gepresst, so dass der Überblick verloren ging.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.71) Es gibt **im** Momente **im** Leben, die alles verändern.	Süddeutsche Zeitung, 19. Februar 2010, S. 20, «Drei Euro für zwei Beutel Lebensmittel» (Hannah Wilhelm)	Doppeltes Auftreten von Wörtern, Präpositionen	Eventuell wurde mit der Stellung von «*Leben*» und «*Momente*» experimentiert.
(A.72) Es **seien keine** jugendtypischen Straftaten und Reifeverzögerungen bei den Angeklagten **seien nicht** erkennbar.	Uckermark Kurier, 26. Februar 2010, S. 14, «Junge aus Wallmow mit Tritten traktiert» (Stefan Adam)	Überflüssige Wörter, Verben	Hier wurde mit zwei verschiedenen Satzkonstruktionen experimentiert, die inhaltlich beide das gleiche aussagen: «*Es seien keine …erkennbar.*» und «*Jugendtypische Straftaten …seien nicht erkennbar.*»
(A.73) Momentan **prüft** die zuständige Staatsanwaltschaft München II **prüft**, ob in Kloster Ettal noch verfolgbare Straftaten vorliegen.	Süddeutsche Zeitung, 25. Februar 2010, S. 2, «Das Kreuz des Schweigens. Die Heimlichtuerei des Ettaler Abts Bögle nährt die Zweifel daran, ob die Kirche Missbrauchsfälle tatsächlich aufklären will» (Heiner Effern)	Doppeltes Auftreten von Wörtern, Verben	Vermutlich wurde hier «*momentan*» später hinzugefügt, die ursprüngliche Fassung könnte gelautet haben: «*Die zuständige Staatsanwaltschaft München II prüft, ob …*».
(A.74) «Übrigens **ist** er **ist** der tollste Vater, den ich mir vorstellen kann, das muss auch mal gesagt werden.»	Der Tagesspiegel, 21. Februar 2010, S. S5, «Interview von Annabel Wahba mit Helene Hegemann»	Doppeltes Auftreten von Wörtern, Verben	Eventuell wurde «*Übrigens*» erst nachträglich eingefügt.
(A.75) Ich habe leider auch keine anderen großen Corpora, **bin** ja arm dran **bin**, da ich ja Koreferenzresolution und Sentiment Detection machen muss.	Manfred Klenner, 16. März 2010, in einer privaten E-Mail[17]	Doppeltes Auftreten von Wörtern, Verben	Vermutlich wurde der erste Nebensatz bereits mit «*da*» eingeleitet, nach Anhängen des zweiten, ebenfalls mit «*da*» eingeleiteten Nebensatzes wurde dies rückgängig gemacht und das Verb korrekt eingefügt. Das ursprüngliche finite Verb ging vergessen.

17. Message ID `<93346320865efb48ae5833dbf6be2360@ifi.uzh.ch>`

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.76) **Bereits** zum Zeitpunkt der Premiere des Streifens waren einige der jugendlichen Darsteller **bereits** an der Front gefallen.	Uckermark Kurier, 3. Mai 2010, S. 23, «Scherze zünden immer noch» (Matthias Bruck)	Doppeltes Auftreten von Wörtern	Eventuell wurde «*Bereits*» aus stilistischen Gründen beim Redigieren an den Anfang des Satzes gestellt, der ursprüngliche Ort dann jedoch nicht verändert.
(A.77) Anders als in «Der gute Hirte aus Schalke» auf Seite 23 vom 26. April geschrieben, **steht** der gute Hirte **steht** nicht in der Offenbarung des Johannes, sondern im Johannesevangelium und zwar in Kapitel 10, Vers 14.	Süddeutsche Zeitung, 7. Mai 2010, S. 33, «Korrekturen»	Doppeltes Auftreten von Wörtern, Verben	Was diesen Fehler besonders interessant macht, ist die Tatsache, dass er in einer redaktionellen Stellungnahme auftritt, in der ein (inhaltlicher) Fehler in einem früher veröffentlichten Artikel korrigiert wird.
(A.78) Gabriela Frei Friedrich und **ihrem Mann** konnten ein ganz spezielles Einfamilienhaus in Grüningen erwerben, das **aus aus** der Feder von Pierre Zoelly stammt.	Neue Zürcher Zeitung, 16. Mai 2010, S. 13, «Auf Gratwanderung» (Anita Simeon)	Doppelte Wörter, Präpositionen, Kongruenz	Hier sind gleich mehrere Fehler zu finden: Die Phrase «*ihrem Mann*» ist Dativ, der Satz würde jedoch verlangen, dass es Nominativ wäre. Im Nebensatz wird die Präposition wiederholt. Zudem ist die Verwendung von «*aus einer Feder stammen*» etwas unglücklich und passte eher zu einem Text oder Bild.
(A.79) A. Z. **passt** sehr gut in das ISAGE-Team **passt**, und sie ist bereits mit einigen Mitarbeitenden der HSA vernetzt.	Protokoll einer Sitzung (Sonja Markwalder), 24. Februar 2010	Doppeltes Auftreten von Wörtern, Verben	Hier wurde vermutlich auch ein längerer Satz aufgetrennt, das finite Verb an die zweite Stelle geschrieben und die ursprüngliche Stelle nicht entfernt.
(A.80) Kurz **nach dem nach dem** Abitur traf ich meinen ehemaligen Direktor im Pressecafé in der Friedrichstraße.	Süddeutsche Zeitung, 10./11. April 2010, S. 14, «Warum ich nie Latein gelernt habe» (Monika Maron)	Doppeltes Auftreten von Wörtern	Hier folgen die doppelten Wörter direkt aufeinander, eine Erklärung bietet allenfalls die Annahme einer Unterbrechung während des Schreibens.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.81) Sagunt sei **ein ein** Bauernopfer gewesen, «das Rom dazu verhalf, unter Wahrung eines Scheins von Rechtmäßigkeit sein strategisches Ziel, den Krieg gegen Karthago, zu erreichen».	Der Tagesspiegel, 20. Mai 2010, S. 28, «Eine Grossmacht wird vernichtet. Nach Vertragsbrüchen löschte Rom 146 v.Chr. Karthago aus» (Michael Zick)	Doppeltes Auftreten von Wörtern, Artikel	Siehe A.80.
(A.82) Für Angela Siebert und Gudrun Fidorra von der Boitzenburger Früchtezauber GmbH ist fast das ganze Jahr lang Saison. In den letzten Tagen haben die «Früchtezauberinnen» vor allem Rhabarber und Holunderblüten geerntet. **Verkauft wird** in den Q-Regio-Läden in Brandenburg und Berlin **verkauft werden**.	Uckermark Kurier, 19./20. Juni 2010, S. 5, «Regionalmarke fehlt es noch an Kraft» (Matthias Bruck)	Doppeltes Auftreten von Wörtern	Denkbar ist, dass der letzte Satz ursprünglich direkt als Nebensatz anschloss: «, *die in den Q-Regio-Läden …verkauft werden*.»
(A.83) Müssen also Menschen, denen die Zeit am Arbeitsplatz lang wird, dafür ungerechterweise **auch noch auch noch** mit einem Abzug vom Lebenszeitkonto büßen?	Der Tagesspiegel, 17. Juni 2010, S. 38, «Zu Tode gelangweilt. Studie: Wer sich im Job nicht ausgefüllt fühlt, verbringt auch seine Freizeit weniger aktiv und senkt so seine Lebenserwartung» (Adelheid Müller-Lissner)	doppeltes Auftreten von Wörtern	Siehe A.80.
(A.84) Die Polizeioperation erfolgte **nach nach** jahrelangen juristischen Querelen, die schliesslich mit einem Urteil des Obersten Gerichts beendet worden waren.	Neue Zürcher Zeitung, 29. Juli, 2010, S. 5, «Beduinendorf in Israel zerstört. Aufeinanderprallen verschiedener Kulturen» (George Szpiro)	Doppeltes Auftreten von Wörtern, Präpositionen	Siehe A.80.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.85) NVP wird dabei vom Vogelpark Marlow und vom Bildhauer Tobias Bork vertreten, der **stellt** Holzkunst **vorstellt**.	Ostsee-Zeitung, 11. August 2010, S. 10, «Kreis ist beim MV-Tag vertreten»	Doppeltes Auftreten von Wörtern, Verben	Möglich wäre «…, *der stellt Holzkunst vor*» oder «…, *der Holzkunst vorstellt*». Erschwert wird das Erkennen des Fehlers durch den Umbruch: Im Original ist «*vor-stellt*» getrennt und der Teil «*stellt*» steht auf einer neuen Zeile.
(A.86) **Der der** Koalitionspartner Linkspartei ist jedoch strikt dagegen, Kinder einzusperren, sondern will sie frühzeitig aus ihren kriminellen Familien herausholen.	Der Tagesspiegel, 31. Juli 2010, S. 1, «Berlins Polizeichef greift Jugendämter an. Glietsch: Kriminelle Kinder schneller aus den Familien nehmen / Neues Konzept für Heime gefordert» (Patricia Hecht, Stefan Jacobs, Ralf Schönball)	Doppeltes Auftreten von Wörtern, Artikel	Siehe A.80.
(A.87) Zum Beispiel **sind** bei der Retrodigitalisierung der Sammlung schweizerischer Rechtsquellen spezielle Verfahren nötig **sein**, um einen effektiven Zugang zu den historischen Texten zu ermöglichen.	Aus einem Forschungsantrag, berichtet von Michael Piotrowski	Doppeltes Auftreten von Wörtern	Ursprünglich war der Satz im Futur («werden nötig sein»), dann sollte dies geändert werden und der Infinitiv wurde vergessen.
(A.88) Er erlitt kurz nach der Attacke mit Schlägen und Triten, die offenbar auch gezielt gegen den Kopf des Opfers gerichtet waren, einen Herzstillstand und starb. Die Jugendlichen **haben** den Übergriff gestanden **haben**, aber eine Tötungsabsicht bestreiten sie.	Süddeutsche Zeitung, 22. Juli 2010, S. 36, «Wer hat angefangen? Im Mittelpunkt des Brunner-Prozesses steht die Frage, von wem die Gewalt ausgegangen ist – die Aussagen dazu widersprechen sich» (Christian Rost)	Doppeltes Auftreten von Wörtern, Verben	Eventuell war der zweite Satz ursprünglich anders formuliert: «*Die Jugendlichen, die den Übergriff gestanden haben, bestreiten eine Tötungsabsicht.*»

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.89) Zudem **gilt** die Pflanze **gilt** als äußerst widerstandsfähig – Regen, Temperaturschwankungen und hohe Luftfeuchtigkeit lassen sie unbeeindruckt.	Uckermark Kurier, 4. August 2010, S. 16, «Edeldisteln erneut geknickt am Boden» (CB)	Doppeltes Auftreten von Wörtern, Verben	
(A.90) Die beiden, für Laien nur als Duo Stubenhocker und Märchensammler denkbar, erweisen sich doch als Typen mit divergierenden Ansichten, was Berufung und Bindungen betrifft. Jacob **noch** ein gutes Stück mehr Kauz **noch** als Wilhelm.	Uckermark Kurier, 4./5. September 2010, S. 31, «Liebeserklärung an die Sprache, zwei Käuze und das Ich» (Roland Gutsch)	Doppeltes Auftreten von Wörtern	Im Gegensatz zum vorherigen Beispiel ist hier das doppelte Wort an beiden Stellen möglich.
(A.91) Ob es **den den** Jungen wirklich gibt oder ob er nur eine Phantasiegestalt ist, spielt keine Rolle.	Uckermark Kurier, 4./5. September 2010, S. 31, «Punktlandung mit Neffe Theo» (Wolfgang Mahlow)	Doppeltes Auftreten von Wörtern, Artikel	Siehe A.80.
(A.92) Das alles sollte man nicht **zu** ernst **zu** nehmen – glaubt Uli Hoeneß.	Süddeutsche Zeitung, 15. September 2010, S. 27, «Prüfung für das van-Gaal-Modell. Der Bayern-Trainer hat auf teure Zugänge verzichtet, ob das richtig war, wird die Champions League zeigen» (Moritz Kielbassa	Doppeltes Auftreten von Wörtern, Infinitiv mit «*zu*»	

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.93) **Es** handelt **es** sich nicht um die Modifikation des Verbs, sondern es ergibt sich eine ganz neue (idiomatische) Bedeutung, weswegen die Bestandteile nicht getrennt werden können.	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Doppeltes Auftreten von Wörtern, Pronomen	Hier wurde der Satzanfang verändert von «*Dabei handelt es sich nicht …*» zu «*Es handelt sich nicht …*»
(A.94) Noch gibt es keinerlei Vermutungen, wer in den neu entdeckten Gräbern bestattet wurde. *In zwei Wochen werden die archäologischen Arbeiten an den Gräbern gestoppt. Dann werden Umbauarbeiten beginnen.* **In zwei Wochen werden die archäologischen Arbeiten an den Gräbern gestoppt. Dann werden umfassende Umbauarbeiten beginnen.** Die Bischofskirche wird bis 2014 für rund 30 Millionen Euro saniert.	Uckermark Kurier, 1. Oktober 2010, S. 21, «40 Gräber unter Hildesheimer Dom entdeckt» (DPA)	Doppeltes Auftreten von Wörtern	Hier handelt es sich vermutlich um einen Kopierfehler.
(A.95) Nach dem Missmanagement im Finanzministerium unter dem damaligen Minister Rainer Speer (SPD) gerät nun der Umgang mit der Affäre durch **Nachfolger Nachfolgers** Helmuth Markov (Linke) in die Kritik.	Tagesspiegel, 17. September 2010, S. 13, «Geheimvertrag gewährte Käufer verspätete Zahlung» (Thorsten Metzner)	Doppeltes Auftreten von Wörtern, Substantive	Korrekt wären die Varianten «*Umgang mit der Affäre durch Nachfolger Helmuth Markov*» oder «*Umgang mit der Affäre des Nachfolgers Helmuth Markov*».

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.96) Trotzdem berichten Sie über ein «Wunder des Alten Testaments» ausgerechnet **auf der auf der** Wissen-Seite.	Leserbrief von Manfred Schleyer (München) in der Süddeutschen Zeitung, 29. September 2010, S. 37	Doppeltes Auftreten von Wörtern, Präpositionen, Artikel	Siehe A.80.
(A.97) Die Kassenprüfer teilen mit, dass die Kasse **wurde** ordnungsgemäß geführt **wurde**.	Protokoll der GMW-Mitgliederversammlung 2010 (Gabi Reinmann), 15. September 2010	Doppeltes Auftreten von Wörtern	Eventuell wurde der erste Teilsatz erst nachträglich eingefügt und es hiess ursprünglich nur: «*Die Kasse wurde ordnungsgemäss geführt.*»
(A.98) Als die Ermittler im Januar 2009 sein Haus und seine Geschäftsräume durchsuchen, fliegt er mit einem seiner Flugzeuge, einer sechssitzigen Piper Malibu, Richtung Florida. **Er** setzt **er** per Funk einen Notruf ab: Während Turbulenzen sei sein Fenster beschädigt worden, er sei verletzt, blute.	Süddeutsche Zeitung, 13. Oktober 2010, S. 25, «Absturz eines Überfliegers. Ein Finanzinvestor, der seinen Tod bei einem Flugzeugunfall vortäuschen wollte, muss nun zehn Jahre ins Gefängnis» (Hannah Wilhelm)	Doppeltes Auftreten von Wörtern, Pronomen	Eventuell waren beide Sätze ursprünglich nur ein Satz und es hiess «*fliegt er …Richtung Florida, setzt er per Funk …*». Durch das Auftrennen in zwei Sätze musste die Struktur verändert werden, das ursprüngliche Pronomen blieb jedoch erhalten.
(A.99) In Abschnitt 2.1.2.1 sind wir auf Taxonomien von Redigieroperationen eingegangen, die mehrheitlich aus der Beobachtung von Änderungen in Texten entwickelt wurden, jedoch nicht die ursprüngliche **Redigierabsicht Absicht** der Autoren berücksichtigen.	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Doppeltes Auftreten von Wörtern, Substantive	Hier sollte «*Absicht*» genauer spezifiziert werden, darum wurde «*Redigierabsicht*» gewählt, das ursprüngliche Substantiv wurde jedoch nicht gelöscht. Hier fällt auch auf, dass das Kompositum, das ja das ursprüngliche Substantiv enthält, vollständig neu geschrieben wurde. Angeboten hätte sich hier auch, das «*A*» von «*Absicht*» in eine Minuskel zu wandeln und dann vorn «*Redigier*» zu ergänzen.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.100) Der Brieffund bedeutet viel für die **Reformationsforschung Forschung**.	Süddeutsche Zeitung, 3. Mai 2010, S. 14, «Luder an Lang» (SZ)	Doppeltes Auftreten von Wörtern, Substantive	Die Entstehung ist vermutlich analog zum vorigen Fehler.
(A.101) Und dort stelle ich mir manchmal vor, wie es wäre, wenn **sie** der schuhgewordene Mädchentraum **sie** auf den Hintern plumpsen liesse.	Blick am Abend, 27. Oktober 2010, S. 35, «Der Schuh machts – wenn die Sohle stimmt» (Flurina Dünki)	Doppeltes Auftreten von Wörtern, Pronomen	Das Pronomen kann an beiden Stellen vorkommen, darf jedoch nur an einer der Stellen vorkommen.
(A.102) **Sind** Hörer in der Lage **sind**, zwischen kontrastiven und nicht-kontrastiven Äußerungen zu unterscheiden?	Anita Steube, Folien zu ihrem Vortrag auf der DGfS-Jahrestagung in Göttingen, 23. Februar 2011 (Folie 35)	Doppelte Auftreten von Wörtern, Verben	Hier wurde eine Fragestellung umformuliert, eventuell von «*Ob Hörer in der Lage sind, …*», um eine Forschungsfrage zu präsentieren.
(A.103) Die Entwickler **von von** *RUSKIN* **stellten** erst beim Ende des Projektes **feststellten**, dass die Autoren ganz andere Funktionen gewünscht und gebraucht hätten.	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Doppeltes Auftreten von Wörtern, Präpositionen, Verben	Die doppelte Präposition lässt sich am ehesten durch einen Unterbruch beim Schreiben erklären. Das mehrfach vorhandene Verb stammt aus einer Umstellung des Satzes.
(A.104) Anschliessend haben sie an einer der C9-League-Universitäten studiert, **heute** besetzen **heute** sie Führungspositionen.	ZS Zürcher Studierendenzeitung, 25. Februar 2011, S. 17, «Made in China», (Clemens Dittrich)	Doppeltes Auftreten von Wörtern	Möglich sind «*…, besetzen heute Führungspositionen*» oder «*…, heute besetzen sie Führungspositionen.*»

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.105) Ihr Aufruf, verfasst am Donnerstag in Brüssel, warnt vor **kaum nicht mehr** umkehrbaren Folgen des Klimawandels.	Süddeutsche Zeitung, 13. November 2009, S. 16, «Sorge um die Arktis» (cris)	Überflüssige Wörter	Vermutlich war «*nicht mehr umkehrbar*» zu negativ und es sollte dann «*kaum mehr umkehrbar*» daraus werden.
(A.106) Nach einer Odyssee durch die Wirren der französischen Geschichte **wurde** es vor zwei Jahren bei einem 85-jährigen Pensionierten gefunden **wurde**, der diese «Reliquie» angeblich seit 1955 bei sich in aller Diskretion wie einen Schatz gehütet hatte.	Neue Zürcher Zeitung, 16. Dezember 2010, S. 24, «Ein Kopf für einen König. Henri IV identifiziert» (rbp)	Doppeltes Auftreten von Wörtern, Verben	Der Anfang des Satzes war ursprünglich wohl ein Nebensatz, hier wurde also ein längerer Satz aufgetrennt.
(A.107) Das führte, als sich die **Universitäten Hochschulen** breiteren Schichten der Bevölkerung öffneten, ohne deshalb deutlich mehr Geld zu erhalten, zu inakzeptablen Studienzeiten und Abbrecherquoten.	Süddeutsche Zeitung, 10. Dezember 2009, S. 15, «Alles für alle!» (Peter Strohschneider)	Überflüssige Wörter	Es wäre möglich, dass hier ein «*und*» fehlt. Vermutlich wurde jedoch «*Universitäten*» durch «*Hochschulen*» ersetzt (möglicherweise auch umgekehrt), aber da der Autor weiter oben im Text von «*Universität und Fachhochschule*» spricht, wollte er wahrscheinlich eher den spezielleren durch den allgemeineren Begriff ersetzen.
(A.108) Kurz nach Auslaufen der Bieterfrist soll Ecclestone erklärt haben, er wolle die schwedische Kultmarke gemeinsam mit der Luxemburger Investmentfirma Genii Capital **zu** retten.	Süddeutsche Zeitung, 9./10. Januar 2010, S. 25, «Ein überraschendes Angebot» (Thomas Fromm)	Überflüssige Wörter, Infinitiv mit «*zu*»	Eventuell hiess es ursprünglich etwa: «*Kurz nach Auslaufen …erklärt haben, die schwedische Kultmarke …zu retten.*» oder «*…erklärt haben, er habe sich vorgenommen, die schwedische …*»

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.109) Jetzt möchte er sich für Olympia **zu** qualifizieren.	NZZ am Sonntag, 3. Januar 2010, S.22, «Turnen im Schnee» (Christof Gertsch)	Überflüssige Wörter, Infinitiv mit «*zu*»	Hier ist das «*zu*» zuviel, es ist gar kein Infinitiv. Denkbar ist, dass der Satz vorher noch vorsichtiger formuliert war, etwa: «*Jetzt möchte er versuchen, sich für Olympia zu qualifizieren.*»
(A.110) Und hier wie da sollte man die aktuellen Auseinandersetzungen um den Bologna-Prozess als Symptom der Überforderung **der von** Universität und Fachhochschule ernst nehmen.	Süddeutsche Zeitung, 10. Dezember 2009, S. 15, «Alles für alle!» (Peter Strohschneider)	Überflüssige Wörter	Wahrscheinlich sollte «*Überforderung der Universitäten und Fachhochschulen*» in «*Überforderung von Universität und Fachhochschule*» geändert werden. Ersteres könnte als zu konkret auf die einzelnen Hochschulen bezogen aufgefasst werden – letzteres dagegen meint klar das System (eine Aussage, mit der alle einverstanden sind, die aber auch dem Tenor des Artikels entspricht).
(A.111) Damit ragt er selbst unter den kräftigen Schwingern heraus. Doch **im Gegensatz im** Training mit den Sumo wird er eher zu den Leichtgewichten gehören.	NZZ am Sonntag, 3. Januar 2010, S. 25, «Der Brocken als Leichtgewicht» (Anja Knabenhans)	Überflüssige Wörter	In der jetzigen Fassung ist «*im Gegensatz*» zuviel. Vielleicht hiess der Satz vorher «*Doch im Gegensatz dazu wird er im Training mit den Sumo eher zu den Leichtgewichten gehören.*» oder «*Doch im Gegensatz zum Training mit den …*»
(A.112) Bisher war **es** bei unseren Übungen das folgende Verfahren üblich: …	Rohfassung eines wissenschaftlichen Artikels von Michael Piotrowski	Überflüssige Wörter	Vermutlich hiess der Satz ursprünglich «*…war es bei unseren Übungen üblich, dass …*» und wurde um die Phrase «*das folgende Verfahren*» ergänzt. Das nun überflüssige Pronomen «*es*» wurde vergessen.
(A.113) Da wir als Sprache **wird** das Deutsche verwenden, …	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Überflüssige Wörter, Verben	Ursprünglich hiess der Satz in einer passiven Form «*Als Sprache wird das Deutsche verwendet.*» Hier sind mehrere Redigieroperationen erfolgt: (a) Der Satz wurde in eine aktive Form gebracht, (b) von einem einfachen Hauptsatz wurde er in einen subordinierten Nebensatz geändert.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.114) Die Teilnehmenden verpflichten sich, **während** im Monat Juni an mindestens der Hälfte ihrer Arbeitstage mit dem Fahrrad zur Arbeit zu kommen.	E-Mail von Felix Steiner vom 11. Mai 2008	Überflüssige Wörter, Präpositionen	Hier wurde an der Formulierung für die Zeitdauer gearbeitet und verschiedene Varianten probiert: «*während eines Monats*», «*im Juni*» u. ä.
(A.115) In elf Projekten **werden** beraten die projektinternen Betreuer andere Personen oder bilden sie aus, …	Tobias Zimmermann, aus einem eigenen Text, 2008	Überflüssige Wörter, Verben	Der Satz wurde als eine Passiv-Konstruktion begonnen. Dann entschied sich der Autor, den Sachverhalt doch in der aktiven Form zu schildern, das «*werden*» wurde dann vergessen.
(A.116) …wenn wir **hätten helfen können**, …Deine Arbeiten …der wissenschaftlichen Öffentlichkeit **zugänglich machen hätten können**.	Michael Piotrowski, aus einem eigenen E-Mail, 2008	Überflüssige Wörter, Verben	Hier wurden mit mehreren Varianten der Kernaussage experimentiert. Möglich sind: «*Wenn wir hätten helfen können, Deine Arbeiten der Öffentlichkeit zugänglich zu machen.*» wie auch «*Wenn wir Deine Arbeiten der Öffentlichkeit hätten zugänglich machen können.*»
(A.117) Am Mittwoch hatte Mussawi **mit auf** dem Teheraner Zentralfriedhof unter starker Präsenz von Sicherheitsleuten die Beerdigung seines Neffen Ali geleitet, der während der Demonstrationen in der Nähe seines Hauses von Unbekannten erschossen wurde.	Süddeutsche Zeitung, 2./3. Januar 2010, S. 9, «Bereit zum Märtyrertod» (Rudolph Chimelli)	Überflüssige Wörter, Präpositionen	Die Präposition «*mit*» ist hier falsch, vermutlich handelt es sich um den Rest einer ursprünglichen Präpositionalgruppe.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.118) Der UBS-Rettung im Herbst 2008, **die sich** nach dem Zusammenbruch der US-Bank **notwendig wurde**, war nicht von Anfang an eine Chance auf Erfolg beschieden gewesen.	20 Minuten online, 17. Januar 2010, «Die Kugel ging nahe am Kopf vorbei» (Philipp Hildebrand)[18]	Überflüssige Wörter, Verben	Korrekte Varianten wären «*Die UBS-Rettung …, die sich nach dem Zusammenbruch der US-Bank als notwendig ergab/herausstellte …*» oder «*Die UBS-Rettung …, die nach dem Zusammenbruch der US-Bank notwendig wurde …*»
(A.119) Der Lehrbeauftragte für Gerichtspsychiatrie an der Uni Zürich hat **mit unter** dem Titel «Fortres» ein 560-seitiges Handbuch für die Risikobeurteilung bei Straftätern entwickelt, …	Tages-Anzeiger, 22. Oktober 2009, S. 17, «Der leidenschaftliche Fussballer, der nie ein Workaholic werden wollte»	Überflüssige Wörter, Präpositionen, Varianten	Beide der Präpositionen sind in diesem Satz möglich, entweder sollte die eine durch die andere ersetzt werden oder der Autor hat beide probiert und sich dann nicht entschieden.
(A.120) Beide Parteien, heißt es aus Lausanne weiter, **haben hätten** bereits je einen Vertreter aus dem geschlossenen Schiedsrichterpool des Cas benannt.	Süddeutsche Zeitung, 12. August 2009, S. 23, «Zur letzten Instanz»	Überflüssige Wörter, Verben, Varianten	Hier wurde vermutlich nachträglich die indirekte Rede eingefügt. Auch hier wären beide Verbformen möglich, um einen korrekten Satz zu erhalten.
(A.121) Nun fällt Rolfes für das Turnier wohl aus, und Hitzlsperger besitzt nicht die Aussichten, bei seinem Heimatverein **auf** die von Löw geforderten Leistungskriterien zu erfüllen.	Süddeutsche Zeitung, 27. Januar 2010, S. 27, «Verletzte Seele» (Philipp Selldorf)	Überflüssige Wörter, Infinitiv mit «*zu*», Präposition	Korrekt wäre die Variante ohne «*auf*»: «*…besitzt nicht die Aussichten, bei seinem Heimatverein die …Leistungskriterien zu erfüllen*». Vielleicht hiess eine frühere Variante: «*…besitzt nicht die Aussichten, bei seinem Heimatverein auf die …Leistung zu kommen.*», was nicht so schön klingt.

18. http://www.20min.ch/finance/news/story/-Die-Kugel-ging-nahe-am-Kopf-vorbei--10138577 (zuletzt besucht am 8.12.2010, 19:27)

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.122) Jede Promotionskommission kann für einen spezifischen Kandidaten jederzeit weitere Arten bewilligen, wie Punkte **zu erworben werden können**.	1. Version des Entwurfs der neuen Doktoratsordnung des Instituts für Computerlinguistik, 3. September 2009	Überflüssige Wörter, Verben	Möglicherweise wurde der Autor während des Schreibens unterbrochen. Eventuell ist es auch der Versuch einer Änderung von «*wie Punkte zu erwerben sind*» zu «*wie Punkte erworben werden können*».
(A.123) Lange **waren haben wir** unsere Büroräumlichkeiten an der Zähringerstrasse 26 im Herzen von Zürich **gehabt**.	Blog der Firma frentix GmbH, 16. August 2009, «Die letzten Tage sind gezählt - frentix zieht um!» (Florian Gnägi)[19]	Überflüssige Wörter, Verben, Varianten	Hier sind zwei Varianten des Satzes vermischt: «*Lange haben wir …gehabt*» und «*Langen waren wir in unseren …*»
(A.124) Um **von** die Variante «*lesen und schreiben*» in «*schreiben und lesen*» zu ändern, …	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Überflüssige Wörter, Präposition, Kongruenz	Ursprünglich hiess der Satz «*Um von der Variante* «lesen und schreiben» *zu* «schreiben und lesen» *zu kommen, …*». Das ist eher umgangssprachlich, also sollte geändert werden zu «*Um die Variante* «lesen und schreiben» *in* «schreiben und lesen» *zu ändern, …*» und die nun überflüssige Präposition blieb stehen.
(A.125) Wie die SZ aus DFB-nahen Kreisen erfuhr, sollen Bundestrainer und Teammanager nur zwei Tage vor der Sitzung am Donnerstag Vertragsofferten erhalten haben, die **binnen von** nur 48 Stunden zu unterzeichnen gewesen sein.	Süddeutsche Zeitung, 5. Februar 2010, S. 35, «Streit um Löw eskaliert» (T.K.)	Überflüssige Wörter, Präposition	Die Variante «*innerhalb von nur 48 Stunden*» wäre korrekt, bei der Verwendung von «*binnen*» entfällt jedoch auch das «*von*».

19. http://blogs.frentix.com/blogs/frentix/2009/08/16/1250423340000.html (zuletzt besucht am 8.12.2010, 19:28)

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.126) Während zum Beispiel Trainer Funkel trotz der Niederlage das «sehr intensive Spiel» lobte, monierte der Torhüter, man müsse «noch mehr **gegen um** den Klassenerhalt kämpfen».	Süddeutsche Zeitung, 8. Februar 2010, S. 31, «Das verbotene A-Wort» (Jörg Marwedel)	Überflüssige Wörter, Präposition	Offenbar wurde an der Wiedergabe der Äusserung des Torwarts herumgebastelt. Wahrscheinlich hat er sich «*gegen den Abstieg*» ausgesprochen, doch – wie der Titel des Beitrags schon nahelegt – darf «*Abstieg*» eben nicht verwendet werden. Das Gegenteil ist dann «*um den Klassenerhalt kämpfen*» und die ursprüngliche Präposition blieb stehen.
(A.127) Zweieinhalb Jahre nach dem Tod ihres Vaters, eines Norwegisch-Professors, soll Hustvedt auf dem Campus eine Rede **zu seinen Ehren Vaters** halten.	Uckermark Kurier, 6./7. Februar 2010, S. 31, «Auf der Suche nach einem unbekannten Teil des Selbst» (Britta Schmeis)	Überflüssige Wörter	Hier wurde offenbar im markierten Bereich mit verschiedenen Varianten experimentiert: «*zu seinen Ehren*», «*zu Ehren ihres Vaters*»
(A.128) **Laut** der ihnen vorliegende Entwurf eines Anwendungserlasses des BMF zeigt wie stümperhaft die Senkung des Mehrwertsteuersatzes für Hotelübernachtungen durchgeführt wurde.	Leserbrief von Anton Glasl (Freising) in der Süddeutschen Zeitung, 18. Februar 2010, S. 43	Überflüssige Wörter, Adverben	«*Laut*» kann an keiner anderen Stelle im Satz verwendet werden, ist aber eventuell ein Überrest einer gestrichenen Wortgruppe.
(A.129) Es gab auch Lustiges: Der ZDF-Parteienforscher Karl-Rudolf Korte hat offenbar mit ZDF-Moderator Theo Koll das Outfit getauscht. Korte **trägt hat** die Haare jetzt rötlich-schön, Koll versucht's mit einer Art toupiertem Blond.	Der Tagesspiegel, 1. Juni 2010, S. 27, «Unser Star in Berlin. Köhlers Rücktritt überrascht die Sender und beherrscht dann die Programme» (Joachim Huber)	Überflüssige Wörter, Verben	Hier wurden verschiedene Varianten probiert.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.130) Sie sind **in auf** einem fremden Netz, es gelten die folgenden max. Gebühren (in CHF)	Automatisches SMS von *Orange*, erhalten am 30. Juni 2010, 10:30	Überflüssige Wörter, Präpositionen	Auch hier wurden vermutlich verschiedene Varianten probiert.
(A.131) Der mittlerweile elfte Kultursommer im Prenzlauer Dominikanerkloster steht in diesem Jahr unter dem Motto «Licht und Schatten». Wie Christin Gaethke, Veranstaltungskoordinatorin des Dominikanerklosters sagt, ist dieses Thema etwas Besonderes, da «es **mit** der Sonderausstellung über Königin Friederike Luise von Preußen eng **miteinander** verbunden ist».	Prenzlauer Zeitung, 3. Juni 2010, S. 17, «Kultursommer in «Licht und Schatten»» (Christoph Schoenwiese)	Überflüssige Wörter, Präpositionen	Hier wurde vermutlich am wörtlichen Zitat redigiert, um es in den Nebensatz einzupassen.
(A.132) Auch die **Garantie** auf kleinere Klassen in der Grundschule (oder Primarschule) werden gesetzlich **garantiert**:…	Süddeutsche Zeitung, 16./17. Juli 2010, S. 2, «Gemeinsam Lernen. Vier Jahre Grund- oder sechs Jahre Primarschule – was für die Hamburger Kinder mitten in den Ferien auf dem Spiel steht» (Ralf Wiegand)	Überflüssige Wörter	
(A.133) **Da** die Elemente einer Substantivgruppe **stimmen** hinsichtlich Kasus, Genus und Numerus **überein**, **daher** können wir die Schnittmenge der Kategorien dieser Elemente verwenden, um die Kategorie der Substantivgruppe zu bestimmen.	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Überflüssige Wörter	Der Beginn des Satzes ist der Beginn eines Nebensatzes, jedoch ist das finite Verb nicht an letzter Position und der nächste Teilsatz schliesst wiederum als Nebensatz an.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.134) Vom Seemannsgericht zur Spezialität: Labskaus **sorgt erlebt** ein Revival in den deutschen Küchen.	Sommer-Magazin der Ostsee-Zeitung, 4. August 2010, S. 1, «Schlemmer-Tipp: Labskaus – eine Spezialität nicht nur für Seeleute»	Überflüssige Wörter, Verben	Eventuell hiess eine frühere Version «*Labskaus sorgt für ein Revival*» oder «*Labskaus erlebt ein Revival*»
(A.135) Berit Feest aus Koserow fotografierte im Januar dieses Jahres bei einer Rundreise **durch in** Namibia.	Uckermark Kurier, 14./15. August 2010, S. 28, Bildunterschrift	Überflüssige Wörter, Präpositionen	Hier wurden beide Varianten probiert.
(A.136) Nach diesem Reformplan **wären sind** dem Inspekteur des Heeres nur noch vier sogenannte Einsatzkommandos in Divisionsstärke und zwei Brigaden unterstellt.	Süddeutsche Zeitung, 9. August 2010, S. 6, «Halbierung des Heeres» (AFP, KNA)	Überflüssige Wörter, Verben	Im Beitrag wird sinngemäss aus einem internen Papier des Verteidigungsministeriums zitiert. Vermutlich wurde also nachträglich der Text daraufhin überprüft, ob alle diesbezüglichen Aussagen entsprechend formuliert sind und darum «*sind*» geändert – wenn auch «*seien*» statt «*wären*» die korrekte Variante wäre, wird doch im vorhergehenden Satz «*ergebe*», also Konjunktiv I verwendet.
(A.137) Die Kredite, mit denen Sigmund seinen Hofstaat, die Beamten und seine Bauten finanzierte, **übergab** Jakob Fugger nicht dem Herzog **aus**, er zahlte sie direkt an die Beamten, Söldnerführer und Handwerker.	Tagesspiegel, 15. August 2010, S. S7, «Jakob Fugger, der Kaisermacher» (Erwin Starke)	Überflüssige Wörter, Verbzusätze	Vermutlich hiess eine erste Fassung «*…zahlte Jakob Fugger nicht dem Herzog aus, er zahlte sie direkt …*» und dann wurde «*auszahlen*» durch «*übergeben*» ersetzt, wobei ersteres ein Verbgefüge ist, letzteres nicht, der Verbzusatz blieb dann übrig.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.138) Das Bild zeigt Anna L. blutverschmiert, kurz nachdem sie ihren Mann Giuseppe L. mit einem Bügeleisen erschlug, weil **diese er sie** angeblich zum Sex zwingen wollte.	Blick am Abend (Ausgabe Basel), 2. September 2010, S. 4, «Das Blut des Gatten» (Ralph Donghi)	Überflüssige Wörter	Hier wurde vermutlich der Satzanschluss mit «*dieser*» und mit «*er*» probiert.
(A.139) Roger Federer hat auch beim zweiten Auftritt am US Open nur wenig Energie verloren. In 101 Minuten bezwang er den Deutschen Andreas Beck 6:3, 6:4, 6:3 **durch**.	20 Minuten (Ausgabe Zürich), 3. September 2010, S. 55, «Nach Blitzstart liess Federer nichts mehr anbrennen» (SI/BT)	Überflüssige Wörter, Verbzusatz	Offenbar hiess eine frühere Variante «*In 101 Minuten setzte er sich gegen den Deutschen …durch.*» Dies wurde dann geändert, da «*bezwingen*» stärker ist als «*durchsetzen*», nur blieb dann der Verbzusatz übrig.
(A.140) Lesen am Bildschirm **ist wird** aus verschiedenen Gründen als unangenehm empfunden:	Christian Schorno, 13.6.2006, Seite «Bildschirmtypografie» in seinem Wiki[20]	Überflüssige Wörter, Verben	Hier wurden verschiedene Formulierungen probiert.
(A.141) So entschloss sich Oppenheim, 1930 in der ehemaligen Freund'schen Maschinenfabrrik in Charlottenburg mit privatem Geld sein Museum **zu aufzubauen**, in dem die Schätze von Guzana ausgestellt wurden.	Tagesspiegel, 3. November 2010, S. 24, «Eine Göttin kehrt zurück. Archäologische Sensation: In Berlin wird der 3000 Jahre alte Schatz aus Tell Halaf zusammengefügt», (Rolf Brockschmidt)	Überflüssige Wörter, Infinitiv mit «*zu*», Präpositionen	Eventuell hiess eine frühere Version «*…sein Museum zu errichten*», dann wurde das ursprünglich vorhandene Verb (eine Ableitung) durch ein anderes, nämlich «*aufbauen*» (ein Verbgefüge), ersetzt. In Verbgefügen wird für den Infinitiv mit «*zu*» das «*zu*» jedoch zwischen Verbzusatz und Verb gestellt – was richtig erfolgt ist –, das ursprüngliche «*zu*» blieb stehen.

20. http://elbanet.ethz.ch/wikifarm/schorno/index.php?n=Main.Bildschirmtypografie (zuletzt besucht am 20.9.2010, 23:35)

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.142) Betrunkene Jugendliche dominieren auch hier die Szene, die Anzahl heimkehrender Partygänger **ist erreicht** den Höchststand.	ZS Zürcher Studierendenzeitung, 25. Februar 2011, S. 12, «Nüchtern in vollen Zügen», (Andreas Rizzi)	Überflüssige Wörter, Verben	Die erste Fassung könnte «*[…], die Anzahl heimkehrender Partygänger ist auf dem Höchststand.*» oder «*[…], die Anzahl heimkehrender Partygänger erreicht den Höchststand.*» gewesen sein.
(A.143) Er wollte das Gulag-System («aus Gründen wirtschaftlicher Ineffizienz») abbauen, den Folterpraktiken **der durch die** Sicherheitsorgane ein Ende bereiten, die Sowjetregierung der westlichen Ukraine, der baltischen Länder und Ostdeutschlands rückgängig machen und das Land vom Stalinkult befreien – ein Programm, das, wie er meinte, seiner eigenen Diktatur weithin Unterstützung verschaffen werde.	Orlando Figes: «Die Flüsterer. Leben in Stalins Russland». (aus dem Englischen von Bernd Rullkötter) Berlin Verlag, Berlin 2008, S. 744f.	Überflüssige Wörter	Vermutlich wurde die Aufzählung redigiert, um sie einheitlich zu gestalten. Korrekt wären «*Er wollte …, den Folterpraktiken durch die Sicherheitsorgane ein Ende bereiten*» und «*Er wollte …, den Folterpraktiken der Sicherheitsorgane ein Ende bereiten*».
(A.144) Erdogan hat nun bekräftigt, was er schon bei seinem fulminanten Abgang vor Jahresfrist angekündigt hatte: dass er in diesem Jahr und vielleicht sogar nie mehr nach Davos kommen werde, wo die Staatenlenker eigentlich die Gelegenheit hätten, in entspannter Atmosphäre Streit ausräumen.	Süddeutsche Zeitung, 13. Januar 2010, S.7, «Ende einer wunderbaren Freundschaft» (Peter Münch)	Verben, Infinitiv mit «*zu*», Fehlende Wörter	Es müsste «*auszuräumen*» heissen.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.145) Es würde uns freuen, Ihre Seminar- und Kongressteilnehmer auch 2009 wieder im Hotel Sternen Oerlikon beherbergen dürfen.	Aus der Vereinbarung über Spezialpreise 2009 für die Universität Zürich mit dem Hotel *Sternen Oerlikon*, 7. Januar 2009	Verben, Infinitiv mit «*zu*», Fehlende Wörter	Eventuell hiess der Satz ursprünglich: «*Es würde uns freuen, wenn wir Ihre Seminar …beherbergen dürfen.*»
(A.146) Als er abgelehnt wird, weil sein Lehrer einen falschen Antrag abschickt, **will** Shaun mit Hilfe seiner Freundin und seines Bruders mit eigenen Mitteln an die Uni **zu kommen**.	Zusammenfassung des films *Nix wie raus aus Orange County* in TV täglich Nr. 47 2009, S. 17	Verben, Infinitiv mit «*zu*»	Wahrscheinlich hiess es ursprünglich «*…will Shaun versuchen, …an die Uni zu kommen*», was dann vermutlich aus platzgründen (es steht nur ein kleiner Kasten zur verfügung) gekürzt werden musste. Das weit entfernte «*zu*» wurde dabei übersehen.
(A.147) Die Rache der Revolutionäre war fürchterlich: George Washington **wollte** die Sechs Nationen von der Landkarte **zu tilgen**.	Süddeutsche Zeitung, 16. Juli 2010, «Das letzte Spiel der freien Irokesen» (Claus Biegert)	Verben, Infinitiv mit «*zu*»	Eventuell war das «*wollte*» ursprünglich «*versuchte*».
(A.148) Ein linientreuer Kandidat soll die Gunst der Stunde nutzen, um die Kernthemen der SVP zu propagieren (Ausländer, EU). Und er **soll** den Abstimmungskampf für die SVP-Ausschaffungsinitiative **zu befeuern**.	Neue Zürcher Zeitung, 17. Juli 2010, S. 11, «Wiedersehen bei Philippi. Die SVP nutzt Bundesratswahlen resolut zur Profilierung - die Kampansage an die SP folgt dem bekannten Schema» (René Zeller)	Verben, Infinitiv mit «*zu*»	Vermutlich waren die beiden Sätze ursprünglich ein Satz, der dann mit «*und*» getrennt wurde. Dabei wurde richtig das Hauptverb «*soll*» wiederholt, jedoch übersehen, dass im ursprünglichen Satz ein Verbgefüge die Prädikatrolle einnahm («*soll die Gunst der Stunde nutzen*»), das den Infinitiv mit «*zu*» verlangt, während «*soll*» nur den Infinitiv verlangt.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.149) Ahrendt **wolle** durch solche «an den Haaren herbeigezogenen und völlig haltlosen Unterstellungen» Aufmerksamkeit im Sommerloch **auf sich zu ziehen**, so Nieszery weiter.	Ostsee-Zeitung, 12. August 2010, S. 6, «FDP-Chef schwärzt Sellering an» (Jörg Köpke)	Verben, Infinitiv mit «*zu*»	Vermutlich stand in einer ersten Version «*versuche*» statt «*wolle*», dann wäre die Fortsetzung mit dem Infinitiv mit «*zu*» korrekt.
(A.150) Gründe dafür **liegen es** auf verschiedenen Ebenen.	Eigener Text, Oktober 2009	Verben	Ursprünglich hiess der Satz: «*Gründe dafür gibt es auf verschiedenen Ebenen.*» Dann wurde «*gibt*» zu «*liegen*» geändert.
(A.151) Im geplanten Beitrag wird argumentiert, dass eine erweiterte Form der Datenerhebung vorgeschlagen, die es ermöglicht, die gewünschten Zusammenhänge zu ermitteln.	Eigener Text, Oktober 2009	Verben	Sinnvoll wäre: «*Im geplanten Beitrag wird eine erweiterte Form …zu ermitteln.*» Weiter oben im Text gibt es einen ähnlichen Textabschnitt, der ebenfalls mit «*Im geplanten Beitrag wird argumentiert, dass …*» beginnt. Offenbar sollte die inhaltliche Ähnlichkeit offensichtlicher gemacht werden.
(A.152) Die organisierte Kriminalität habe sich die Tatsache zunutze gemacht, dass Politiker, Zoll- und Geheimdienstbeamte sowie staatliche Inspektoren in Pilsen **studierten begonnen hätten**, weil sie nach einer neuen Gesetzgebung eine akademische Ausbildung brauchten, um ihre Posten zu behalten.	Süddeutsche Zeitung, 21. Oktober 2009, S. 10, «Dr. Schummel»	Verben	Das hätte vermutlich «*zu studieren begonnen hätten*» werden sollen und könnte ursprünglich «*dass …Inspektoren in Pilsen studierten, weil …*» geheissen haben.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.153) Von 1941 bis 1955 **wurden** von diesen Gerichten insgesamt 70 000 Deutsche einerseits wegen Kriegsverbrechen und andererseits aus politischen Gründen, die im Zusammenhang mit dem Aufbau des stalinistischen Besatzungsregimes standen, **verurteilt worden**.	Uckermark Kurier, 29. Dezember 2009, S. 3, «Justiz der Besatzer»	Verben, Tempus	Entweder müsste es heissen «*Von 1941 bis 1955 wurden …verurteilt.*» oder «*Von 1941 bis 1955 sind …verurteilt worden.*»
(A.154) Mit entwaffnender Offenheit gibt er zu, er **habe** immer die Maske des Entertainers **getragen** und nie echt **gewesen zu sein**.	Tages-Anzeiger, 24. November 2009, S. 35, «Ein enttäuschter, gekränkter Aussenseiter» (Claudia Kühner)	Verben, Modus	Der erste Teil des Nebensatzes ist indirekte Rede, der zweite Teil berichtet eine Tatsache. Vermutlich wurde entweder der erste Teil noch eingeschoben oder der zweite später ergänzt.
(A.155) Dem Buchstaben nach **oblag** die Aufsicht über den Fonds **bei** einem Organ, das sich paritätisch aus zwei Vertretern des erweiterten Rektorats sowie je einem Vertreter der St. Galler Studentenschaft und des damaligen Muttervereins des Forums HSG zusammensetzte.	Neue Zürcher Zeitung, 19. Juli 2010, S. 33, «Job-Messen sind Chefsache. An der HSG gibt es keinen Platz mehr für einen rein studentischen Rekrutierungs-Event» (Robin Schwarzenbach)	Verben, Präposition	Die Formulierung «*obliegen bei (+ Dativobjekt)*» existiert nicht, möglich ist hingegen «*liegen bei (+ Dativobjekt)*» oder «*obliegen (+ Dativobjekt)*».
(A.156) Überdies **gelte** längeres gemeinsames Lernen eher **dem herrschenden Standard** in Europa.	Süddeutsche Zeitung, 17./16. Juli 2010, S. 2, «Gemeinsam Lernen. Vier Jahre Grund- oder sechs Jahre Primarschule – was für die Hamburger Kinder mitten in den Ferien auf dem Spiel steht» (Ralf Wiegand)	Verben	Hier wurde offenbar mit verschiedenen Verben experimentiert. Korrekt wären folgende Varianten: «*Überdies gelte längeres gemeinsames Lernen als Standard*», «*Überdies entspreche längeres gemeinsames Lernen her dem Standard*».

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.157) Laut Polizei hatte die Frau gegen 10.50 Uhr zu Fuß die Kreuzung in Höhe Platz der Freundschaft überqueren wollen, dabei aber **nicht auf** die heranfahrende Straßenbahn der Linie 5 **nicht bemerkt**.	Ostsee-Zeitung, 4. August 2010, S. 4, «Frau nach Unfall mit Strassenbahn schwer verletzt»	Verben	Möglich wäre entweder «…, *dabei aber nicht auf die Straßenbahn geachtet*» oder «…, *dabei aber die Straßenbahn nicht bemerkt*»
(A.158) Ich **möchte mich** allen **danken**, die mir das ermöglichen, vor allem meinen Eltern, die seit zehn Jahren mich und meinen Sport unterstützen, meinem erten Trainer Uwe Monser, der mir das Segeln und den Wettkampfsport beibrachte, und meinem neuen Verein, dem BSV, der mich tatkräftig unterstützt.	Ostsee-Zeitung, 6. August 2010, S. 16, «Barther Segler jetzt in Nationalmannschaft» (Interview: HJM)	Verben	Korrekt wäre die Variante «*Ich möchte mich bei allen bedanken, die mir …*» oder «*Ich möchte allen danken, die mir …*». Die zweite Variante ist offenbar die zuletzt editierte, da die Aufzählung nur zu dieser passt.
(A.159) «In der Bundesliga hat jetzt eine stabile zu Phase kommen», fordert Rummenigge	Süddeutsche Zeitung, 24./25. Oktober 2009, S. 35, «Schon wieder ein neuer Trainer»	Wortstellung	Hier wurde der Autor vermutlich unterbrochen. Allenfalls ist die falsche Wortstellung auch durch Kopieren und Einfügungen entstanden.
(A.160) «Ich habe die aktuelle sportliche Situation heute mit Rainer Adrion erörtert», sagte Löw, und aus sportlichen Gründen sie hätten sich darauf veständigt, dass Thomas Müller bei der U21 bleibe.	Süddeutsche Zeitung, 13. November 2009, S. 25, «Grippeverdacht bei Klose» (sid)	Wortstellung	Eventuell waren das ursprünglich zwei Sätze, das Zitat von Löw und die Verständigungt mit Thomas Müller, oder beide waren von Beginn an mit Komma verbunden: «*…sagte Löw, sie hätten sich darauf verständigt …*». Und dann mussten die sportlichen Gründe noch untergebracht werden.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.161) Die Shrimps sind verdorben, so hat sich Hermann Walzer (Armin Rohde, r., mit Uwe Ochsenknecht) das Hochzeitsbankett für seine Tochter nicht vorgestellt – und verweigert prompt zu die Rechnung zahlen.	Süddeutsche Zeitung, 2./3. Januar 2010, S. 40, Bildunterschrift zur Ankündigung des Films «Bluthochzeit»	Wortstellung, Infinitiv mit «*zu*»	Siehe A.159.
(A.162) Spekulationen, wonach dies ein Teil eines Deals mit dem Autor Airen – der Blogger weiterhin anonym bleiben will – sein könnte, kommentierte Ullstein nicht.	Uckermark Kurier, 23. Februar 2010, S. 21 (Dirk Schroeder)	Wortstellung	Der Einschub ist ein Hauptsatz, das finite Verb müsste daher an zweiter Stelle stehen, nicht am Ende. Vermutlich war dieser Einschub vorher ein relativierender Nebensatz.
(A.163) So Eritrea gibt mehr als 20 % seines Bruttosozialprodukts für das Militär aus.	Blick am Abend (Ausgabe Zürich), 15. März 2010, S. 13, «Waffen für die Dritte Welt»	Wortstellung	Der Satzanfang «*So*» wurde vermutlich erst später hinzugefügt und hätte eine Umstellung erfordert.
(A.164) Der Verein **lang finanzierte** Fortbildungen für Wirtschaftsjournalisten.	Süddeutsche Zeitung, 10./11. April 2010, S. 34, «Der Mann im Schatten» (Uwe Ritter)	Wortstellung	Hier sind Adverb und finites Verb vertauscht worden.
(A.165) Die kantonale BVG- und Stiftungsaufsichtsbehörde **sodann trat** auf den Plan mit dem beruhigenden Hinweis, 8,2 % aller Kassen hätten Ende 2008 mit einer Unterdeckung von 90 % bilanziert.	Neue Zürcher Zeitung, 19. Mai 2010, S. 26, «Langes Warten auf schlüssige Statistiken» (nz)	Wortstellung	Hier wäre ebenfalls möglich, dass «*sodann*» erst später (und an der falschen Stelle) eingefügt wurde.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.166) Von der grauhaarigen Frau sieht Christin nur **zuckenden den** Rücken.	Wim Westfield: «Nur Engel fliegen höher». (Roman), edition q im be.bra verlag GmbH, Berlin-Brandenburg 2008, S. 13.	Wortstellung	Siehe A.159.
(A.167) Wie ein Häufchen Elend saß Messi in der Kabine, weinte hemmungslos, war von niemandem zu trösten. Wortlos und leichenblass schlich er eineinhalb Stunden **dem nach** Spielende aus dem Stadion «Green Point» in Kapstadt.	Uckermark Kurier, 5. Juli 2010, S. 16, «Vernichtendes Urteil über den Floh: Messi ist nicht Maradonna. Der argentinische Ausnahmespieler ist eine der grossen Enttäuschungen der WM in Südafrika» (Marco Mader)	Wortstellung, überflüssige Wörter	Die Variante ohne «*dem*» wäre korrekt, eventuell wurde es später eingefügt und dies geschah an der falschen Stelle.
(A.168) Entlang der Glatt brauche **aber es** weder einen verbesserten Hochwasserschutz noch ein neues Naturschutzgebiet, heisst es in der Mitteilung weiter.	Neue Zürcher Zeitung, 25. Mai 2010, S. 13, «Oberglatt gegen Glatt-Revitalisierung» (hhö)	Wortstellung	Auch hier ist möglich, dass «*aber*» erst später eingefügt wurde.
(A.169) Postfahrzeug rollt **in Teich einen**	Uckermark Kurier, 6. September 2010, S. 4, Überschrift (dpa)	Wortstellung	
(A.170) Natürlich greifen bildungsnahe Elternhäuser ein und schließen **die Lücken Wissen** ihrer Sprösslinge schon in den ersten Jahren.	Süddeutsche Zeitung, 22. September 2010, S. 31, Leserbrief von Nathalie Saab «Benachteiligt sind Kinder bildungsferner Eltern»	Wortstellung	Eventuell hiess es «*die Lücken im Wissen ihrer Sprösslinge*», eigentlich würde man jedoch das Kompositum «*die Wissenslücken*» erwarten.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.171) Versuchen Sie nicht, Ihren **zu Partner verändern**.	Blick am Abend, 1. Oktober 2010, S. 33, Horoskop «Skorpion»	Wortstellung	Die falsche Wortstellung ist wahrscheinlich durch Fehler beim Kopieren und Einfügen entstanden.
(A.172) Mit dem ewig nackten Oberkörper des Olympiers demonstrierte noch der hochbetagte Picasso auf den Bühnen der Badestrände zwischen Cap Ferrat und Saint-Tropez unübersehbar seine anscheinend niemals versiegende Schaffenskraft, als Maler **als wie** Vater.	Süddeutsche Zeitung, 4. Oktober 2010, S. 14, «Gott in Badelatschen: Ein Fotoband folgt Picasso an die Riviera» (Manfred Schwarz)	Wortstellung	Hier ist ebenfalls möglich, dass «*als*» erst später eingefügt wurde.
(A.173) Nur vor diesem Hintergrund konnte Leyens Losung verfangen, dass nach dem Verfassungsgerichtsurteil nicht unbedingt mehr Geld fällig sei, sondern Sachleistungen. Ihr Bildungspaket verspricht einen «Durchschnittswert», den tatsächlichen Nutzen für **zwei Millionen die Kinder** kann niemand beziffern.	Tagesspiegel, 3. Oktober 2010, S. 25, «Wer nicht arbeitet, soll nicht essen» (Tissy Bruns)	Wortstellung	Wahrscheinlich gab es die Zahlenangabe ursprünglich nicht, sondern nur die Formulierung «*Nutzen für die Kinder*». Dann sollte die Anzahl untergebracht werden, was ein Ersetzen von «*die*» erfordert hätte.
(A.174) Wundern Sie sich nicht, **wenn man Sie** aufgrund dieser Sache nun häufiger angesprochen und um Rat **gebeten werden**.	20 Minuten, 26. Januar 2010, S. 20, Horoskop «Skorpion»	Kongruenz, Genus Verbi	Hier wurde vermutlich von einer unpersönlichen zu einer persönlichen Formulierung gewechselt.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.175) …sind **nicht viele neuen** Erkenntnisse hinzugekommen.	Eigener Text von Eva Strübin	Kongruenz	Ursprünglich hiess die Phrase «*sind keine neuen Erkenntnisse hinzugekommen*», der Ersatz von «*keine*» durch «*nicht viele*» erfordert die Anpassung des Adjektivs.
(A.176) Das Projektteam innerhalb des vorliegenden Projekts setzt sich zusammen aus insgesamt fünf wissenschaftlichen Mitarbeitern (drei deutsche und zwei polnische Wissenschaftler), zwei Verwaltungsangestellten (anteilig; jeweils eine/ein Deutsche/Deutscher und eine/ein Pole/Polin), **mehreren polnische und deutsche studentische Hilfskräften** sowie Java- bzw. C/C++-Programmierer aus beiden Ländern.	Aus einem Projektantrag	Kongruenz	Hier ist wohl ein Übertragungsfehler passiert, alle Aufzählungen in Klammern sind jeweils im Nominativ.
(A.177) Bitte schreiben Sie Ihren Beitrag so, **dass er** für ein breites Publikum **verständlich sind**.	Berichtet von Michael Piotrowski, erhalten in einer Rückmeldung über die Annahme eines wissenschaftlichen Beitrages.	Kongruenz	Hier wurde die allgemeine Fassung «*Bitte schreiben Sie Ihre Beiträge so, dass sie für ein breites Publikum verständlich sind.*», gerichtet an Autoren, die einen Artikel für ein Journal einreichen möchten, in eine Fassung an einen konkreten Autor bezogen auf dessen konkreten Artikel geändert. Die Änderung von «*Beiträge*» in «*Beitrag*» inklusive Anpassung der Anapher im folgenden Nebensatz und dem vorangehenden Possessivpronomen war erfolgreich. Dass Subjekt und Verb im Nebensatz nicht mehr kongruent sind, wurde nicht bemerkt.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.178) Sei G ein Graph und f eine Abbildung …	«Vereinheitlichte Darstellung von Techniken zur effizienten Kürzeste-Wege-Suche», Dissertation von Stefan Lewandowski, 2005	Kongruenz	Hier wurde «*und f eine Abbildung*» nachträglich ergänzt, aber versäumt, «*Sei*» in «*Seien*» zu ändern.
(A.179) **Sie lesen** den Newsletter von Chreis3 **und freuen uns** über Ihr Interesse an unseren Produkten. Wir haben viele Neuerungen aufgeschaltet und freuen uns über Ihr Interesse an den Produkten von Chreis3.	Newsletter von *Chreis3* vom 8. Mai 2008	Kongruenz	Hier wurde offenbar mehrfach redigiert. Zwei Hauptsätze sind mit «*und*» verknüpft worden, beim zweiten wurde das Subjekt weggelassen. Dies darf jedoch nur dann geschehen, wenn es mit dem Subjekt des ersten Satzes identisch ist. Das ist hier nicht der Fall, da «*Sie freuen uns über Ihr Interesse …*» ungrammatisch ist. Zum anderen ist der zweite Teilsatz des zweiten Satzes bis auf die Formulierung bezüglich der «*Produkte*» identisch: «*an unseren Produkten*» vs. «*an den Produkten von Chreis3*». Hierfür gibt es mehrere Möglichkeiten der Entstehung: Einer der beiden Sätze war das Original und wurde kopiert, neu eingefügt und leicht verändert, ohne den ursprünglichen Satz zu entfernen; beide Sätze wurden aus verschiedenen Vorlagen kopiert und hier eingfügt.
(A.180) Statt sich darin zu gefallen, die knappen polizeilichen Mittel tagsüber zur Verteilung kleinlicher Parkbussen zu verschwenden, müssten nachts Patrouillen dafür sorgen, dass **die Bevölkerung** in Schwamendingen eine geschützte Heimat **haben**.	Wahlkampfflyer der SVP im Kreis 12 der Stadt Zürich, Februar 2010	Kongruenz	«*Bevölkerung*» bezeichnet natürlich eine Pluralentität, grammatikalisch handelt es sich jedoch um den Singular. Eventuell war ursprünglich von «*Einwohnern*» die Rede.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.181) …und hoffe, dass Sie **einen schöne und erholsame Ferien** hatten.	Eigenes E-Mail von Michael Piotrowski, 2008	Kongruenz	Anfangs wurde das Wort «*Urlaub*» verwendet, das dann durch «*Ferien*» ersetzt wurde. Die Adjektive wurden angepasst, der nun überflüssige Artikel dann vergessen.
(A.182) Funktionen können **auf einem konkretes linguistisches Element** – d. h. **eine konkrete Wortform** …oder **eine konkrete Phrase** …– operieren, auf Wortarten wie Konjunktionen, auf bestimmten Kategorien …	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Kongruenz	Hier wurde redigiert, bevor der Satz beendet wurde. Ursprünglich hiess der Anfang: «*Funktionen können auf ein konkretes linguistisches Element – d. h. eine konkrete Wortform …zugreifen, auf Wortarten wie Konjunktionen*». Dann wurde «*zugreifen auf*» in «*operieren auf*» geändert, die Präposition bleibt die gleiche, verlangt jedoch jetzt den Dativ statt Akkusativ. Nach der Präposition wurde der Artikel angepasst, und im Rest des Satzes wurde beim Schreiben auf die Kongruenz geachtet.
(A.183) Was kann man dann tun, wenn man ein naseweiser, vielleicht etwas altkluger und nach Geltung gierender Teenager ist, der im Kulturbetrieb etwas erreichen will – was kann man tun, außer abschreiben, im Internet, **im einem Roman** eines weitgehend unbekannt gebliebenen Autors mit dem Pseudonym Airen, in einem Kurzfilm von Benjamin Teske oder wo auch immer?	Süddeutsche Zeitung, 11. Februar 2010, S. 15, «Ich bin in Berlin. Es geht um meinen Wahn» (Thomas Steinfeld)	Kongruenz	Hier ist der Grund sicher in stilistisch motiviertem Redigieren zu suchen: Die Aufzählung beginnt mit «*im …*», vermutlich hiess die Fortsetzung «*im Roman eines …, im Kurzfilm von …*». Da jedoch nicht auszuschliessen ist, dass beide Autoren mehr als einen Roman bzw. einen Kurzfilm verfasst haben, ist es nicht möglich, «*im*» (also «*in dem*») zu verwenden.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.184) Rohfassung **eines Wissenschaftlicher Artikel** von …	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Kongruenz	Ursprünglich hiess der Text «*Wissenschaftlicher Artikel von …*» und war an mehreren Stellen im Text platziert, aus stilistischen Gründen wurden diese Textstellen (und ähnliche) jeweils um «*Rohfassung eines*» bzw. «*Rohfassung einer*» ergänzt. Da diese Operation mittels `einfügen` mit der Maus beim Durchgehen durch den Text ausgeführt wurde, ging vergessen, dass die folgende Substantivgruppe noch hätte angepasst werden müssen
(A.185) Die Passanten **hätten** unverzüglich (und vorbildlich) die Ordnungshüter **alarmierten**, welche sich des Mädchens annahmen.	Onlineausgabe von 20 Minuten, 3. Februar 2010, 17:22, «Fünfjährige hielt Polizei auf Trab» (Joel Bedetti)[21]	Zeitform, Verben	Hier wurde mit verschiedenen Zeitformen experimentiert, möglich sind: «*Die Passanten …alarmierten*» oder «*Die Passanten hätten …alarmiert*».
(A.186) Alle Studierenden dokumentieren und reflektieren **die vorgegebene Kompetenzen**.	Rohfassung eines wissenschaftlichen Artikels, Elisabeth Müller, Februar 2010	Kongruenz	Auf den ersten Blick könnte es sich auch um einen Tippfehler (ein vergessenes oder nicht richtig gedrücktes `n` in «*vorgegebene*» handeln. Da der Artikel jedoch mit MS Word unter Verwendung der Funktion `Änderungen nachverfolgen` erstellt wurde, kann die Entstehung eindeutig rekonstruiert werden: Ursprünglich hiess der Satz «*Alle Studierenden dokumentieren und reflektieren fünf vorgegebene Kompetenzen.*» Aus inhaltlichen Gründen sollte die Anzahl der Kompetenzen nicht genannt werden, darum wurde «*fünf*» zu «*die*» geändert – und die erforderliche Anpassung der Substantivgruppe wurde in dieser expliziten Redigierphase, in der der Text «*korrekturgelesen*» wurde, nicht berücksichtigt.

21. http://www.20min.ch/news/basel/story/14657372 (zuletzt besucht am 8.12.2010, 19:28)

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.187) **Eine Wortergänzungssoftware vervollständigen** noch nicht zu Ende geschriebene Wörter, indem der Benutzer das gewünschte Wort aus einer Liste möglicher vollständiger Wörter auswählt.	Seminararbeit von Damian Hiltbrand, November 2009	Kongruenz	Vermutlich war das Subjekt in einer ersten Fassung im Plural.
(A.188) Nach der Pfahls´s gestriger Darstellung die in seinem eigenen Verfahren auch von Ex-Bundeskanzler Helmut Kohl bestätigt worden war, war der heute 67-Jährige nie einflussreich genug, um über Projekte zu entscheiden, für die Schreiber als Lobbyist arbeitete.	Der Tagesspiegel, 23. Februar 2010, S. 4, «Wiedersehen in Augsburg. Im Prozess gegen den Waffenlobyisten Schreiber sagte Ex-Staatssekretär Pfahls aus – die Anklage bröckelt» (Lars von Törne)	Kongruenz	Korrekte Varianten sind «*Nach Pfahls' gestriger Darstellung*» oder «*Nach der gestrigen Darstellung von Pfahls*». Hier kommt noch dazu, dass der Genitiv von Wörtern auf «*-s*» zwar mit Abostroph geschrieben wird, das Genitiv-s dafür jedoch entfällt und nicht etwa nach dem Abostroph noch angefügt wird.
(A.189) Als die Soldaten in den Weltkriegen am großen Morden zweifelten und verzweifelten, bogen sich christliche Feldprediger in ihrer Not das fünfte Gebot zurecht. Zwar lautet es unmissverständlich «Du sollst nicht töten»; aber **die Prediger nahmen** den Krieg vom Verbot aus und **tat** so, als **sei** er eine Parteinahme Gottes.	Süddeutsche Zeitung, 11. August 2010, S. 4, «Töten auf Kommando» (Heribert Prantl)	Kongruenz	Hier war offenbar ursprünglich von «*dem Prediger*» im Singular die Rede oder von «*der Geistlichkeit*».
(A.190) Ihre Arbeitet fertigt sie in Öl und mit Pastellfarben.	Flyer des Dominikanerklosters Prenzlau (D), Februar 2010, Ankündigung einer Ausstellung	Kongruenz, Satzstruktur	Wahrscheinlich hiess eine erste Fassung «*Sie arbeitet mit Pastellfarben*» und «*Ihre Arbeiten fertigt sie in Öl*».

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.191) **Der vor der Pensionierung stehende Beamte sei** aber im Zuge der Überprüfung der Zugang zu Verschlusssachen entzogen worden.	Uckermark Kurier, 22. März 2010, S. 5, «Abhörtechnik beschäftigt Landtag» (Alexander Fröhlich)	Kongruenz	Es müsste «*Dem vor der Pensionierung …*» heissen.
(A.192) Aus diesen Daten ist jedoch nicht eindeutig zu ermitteln, **ob eine Pause** die Pause vor einer Operation oder dem Schreiben eines Wortes oder nach einer solchen Operation **erfolgen**, also eher dem Planen oder dem Redigieren zuzuordnen **sind**.	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Kongruenz	Ursprünglich hiess der Text «*…, ob Pausen vor …erfolgen, also eher dem Planen oder dem Redigieren zuzuordnen sind.*» Dann wurde entschieden, eine Formulierung wie «*ob eine Pause die Pause vor oder nach einer Operation ist*», jedoch wurde nur der erste Teil dieser Änderung umgesetzt.
(A.193) Diese Editierfunktionen sind keine Umsetzung des heute Machbaren unter Berücksichtigung der Qualität und Abdeckung **der verfügbaren computerlinguistischer Ressourcen**, sondern ganz klar dem Ziel verpflichtet, die kognitive Belastung von Autoren zu reduzieren.	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Kongruenz, Adjektive	Ursprünglich hiess der Text «*…Qualität und Abdeckung verfügbarer computerlinguistischer Ressourcen, …*», dann wurde der bestimmte Artikel eingefügt und lediglich das erste Adjektiv hinsichtlich Kongruenz angepasst.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.194) Wer einmal bei **seinem Textverarbeitung** die Rechtschreib- oder gar Grammatikprüfung angeworfen hat, ist sicher so unangenehm überrascht, dass er das nächste Mal gleich die Finger davon läßt.	Thomas Hanneforth: «Computerlinguistik. Computer lernen unsere Sprache», online [22]	Kongruenz	Vermutlich hiess es ursprünglich «*bei seinem Textverarbeitungsprogramm*».
(A.195) Du hast die Beiträge nachvollziehbar aufgegliedert und auch jeweils gleich eine Foto dazugestellt hast, konnte ich alles gut verfolgen.	Rückmeldung einer Studentin auf ein Lernjournal einer Kommilitonin	Satzstruktur, Verben	Der zweite Teil des Satzes ist die Fortsetzung eines anderen Satzes, nämlich «*Da Du die Beiträge …*».
(A.196) …, dann beschied sie schmallippig, in erster Linie schaue sie nun auf sich selbst, «und da steht **die erst mal Enttäuschung** im Vordergrund».	Süddeutsche Zeitung, 19. Februar 2010, S. 27, «Endlich am Ziel» (Wolfgang Gärner)	Wortstellung	Vermutlich ist die falsche Wortstellung durch Kopieren und Einfügen entstanden.
(A.197) Fachblätter für Großanleger sind, versorgen ihre Kundschaft gern mit Checklisten, in denen vor raffinierten Frage- und Recherchetechniken der Steuerermittler gewarnt wird.	Süddeutsche Zeitung, 5. Februar 2010, S. 7, «Wenn die eifersüchtige Gattin hilft» (Katja Riedel, Hans Leyendecker)	Satzstruktur	Eventuell wurde «*versorgen ihre Kundschaft gern mit*» eingeschoben in den ursprünglichen Satz «*Fachblätter für Großanleger sind Checklisten, in denen …*».

22. http://www.ling.uni-potsdam.de/index.php/de/studium/bachelorstudiengaenge/bsc-computerlinguistik/was-ist-computerlinguistik.html (zuletzt besucht am 8.12.2010, 19:42).

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.198) Momentan **sind** die sechs führenden CAT-Systeme auf dem Markt **sind** Across, Trados, Transit, Déja Vu, SDLX und Wordfast auf Grund ihrer unterschiedlicher Einsatzmöglichkeiten.	Aus einer studentischen Arbeit, Herbstsemester 2009	Doppeltes Auftreten von Wörtern, fehlende Wörter, Verben	Im ersten Teilsatz tritt «*sind*» doppelt auf, dafür fehlt im zweiten Teilsatz das Hauptverb.
(A.199) Viele Hausärzte **sähen sich** zeitlich, materiell und körperlich **nicht mehr in der Lage, dem Anspruch**, ein guter Hausarzt sein zu wollen und dem Patienten auch noch sein Ohr zu leihen, **nicht mehr gewachsen**.	Uckermark Kurier vom 19. Februar 2010, S. 17, «Immer mehr Hausarztpraxen schliessen» (Sigrid Werner)	Kollokation	Hier wurden zwei verschiedene Varianten probiert: «*sähen sich …nicht mehr in der Lage, dem Anspruch …zu genügen*» und «*sind, dem Anspruch …nicht mehr gewachsen*».
(A.200) Aus Verlagskreisen ist zu erfahren, dass Ullstein nun SuKuLTur eine kleine vierstellige Summe und wird ab Herbst «Strobo» in Lizenz als Taschenbuch herausgeben.	Süddeutsche Zeitung, 13./14. Februar 2010, S. 13, «Der Schattenmann» (Thorsten Schmitz)	Fehlende Wörter, Kongruenz, Verben	Eine Erklärung für den Zustand dieses Satzes ist die Verschmelzung von zwei Sätzen zu einem: «*Aus Verlagskreisen …, dass Ullstein …eine …Summe zahlen wird. Ullstein wird ab Herbst …herausgeben.*»
(A.201) Und davon, dass zwei Lesben und ein Schwuler zu dritt ein Kind zu bekommen – während die Politik noch streitet, ob sie zu zweit eins adoptieren dürften.	Süddeutsche Zeitung, 20./21. März 2010, S. 14, «Eine Mami und eine Mama» (Charlotte Frank)	Satzstruktur, Infinitiv mit «*zu*»	Eventuell hiess eine frühere Version «*Und davon, als zwei Lesben …ein Kind zu bekommen …*». Der vorhergehende Satz enthält jedoch die Struktur «*Die Geschichte davon, dass …*», so dass der nachfolgende Satz vermutlich angepasst wurde, um die gleiche Struktur zu enthalten.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.202) In diesem Jahr reist der Mecklenburger Buchverlag (MBV) mit vier Publikationen nach Leipzig, und angesichts der mittlerweile eingegangenen Manuskripte in ziemlich kurzer Zeit ein Dutzend weitere hinzukommen, berichtet Verlagsleiter Silvio Pankratz. So eilig will er es nicht angehen, lieber wenige ausgewählte Projekte realisieren.	Uckermark Kurier, 24. Februar 2010, S. 21, «Vier Titel am eigenen Stand» (Susanne Schulz)	Wortstellung, Satzglieder, Fehlende Wörter, Verben	Wahrscheinlich wurde hier ein Nebensatz eingefügt – entweder neu geschrieben oder hineinkopiert – dabei wurde werder auf Interpunktion, noch auf das Verb geachtet, noch auf korrekten Satzanschluss.
(A.203) Der Verband unabhängiger Schweizer Hochzeitsplaner (VSUH) **bietet** einen Diplomlehrgang zum Hochzeitsplaner **durch**.	20 Minuten, 22. März 2010, S. 43, «Eine Hochzeit im Detail planen»	Verben, Verbzusatz, Verbgefüge	Hier wurde höchstwahrscheinlich «*durchführen*» durch «*anbieten*» ersetzt. Beide Verben haben einen Verbzusatz, dieses wurde nicht getauscht.
(A.204) Mit einem Gewinn von 2,2 Milliarden Franken im ersten Quartal hat **die UBS-Chef Oswald Grübel** sogar seinen **ehamligen** Arbeitgeber **schlagen**.	Blick am Abend (Ausgabe Zürich), 4. Mai 2010, S. 8, «Milliardengewinn für UBS – trotz Geldabfluss» (map)	Verben, Kongruenz, Rechtschreibung	Dieser Satz ist gar nicht mehr verständlich.
(A.205) Ihre Haut ist von blutigen Ritzern übersät. **zuvor entführt worden.** Jemand hat ihre Nervenbahnen durchschnitten.	Uckermark Kurier, 24./25. April 2010, S. 31, «Wenn die Lektüre zum Albtraum wird.» (Susanna Gilber-Sättele) Rezension zum Roman «Ausgelöscht» von Cody McFadyen	Überflüssige Wörter	Hier sind drei völlig zusammenhanglose Wörter zwischen zwei syntaktisch korrekten Sätzen übrig geblieben. Sie können aus Umstellungs und Verschmelzungsoperationen vorher vorhandener Sätze übrig geblieben sein, oder sie wurden von der Autorin notiert, um später ergänzt zu werden, was dann nicht mehr erfolgte.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.206) Das ISAGE **wird** an der Tagung des European Centre for Clinical Social Work (ECCSW) in Berlin, 14./25. September 2010, Klinische Sozialarbeit: «Soziale Gesundheit stärken» gemeinsam **teilzunehmen**.	Protokoll einer Sitzung (Sonja Markwalder), 24. Februar 2010	Kongruenz, Verben, Infinitiv mit «*zu*»	Einerseits sollte hier wohl zum Ausdruck gebracht werden, dass das ISAGE-Team an der Tagung teilnimmt. Andererseits ist die Formulierung (im zweiten Teilsatz) in der «wir»-Form, um zum Ausdruck zu bringen, dass das Team beabsichtigt, gemeinsam teilzunehmen. Die folgenden Sätze in diesem Abschnitt sind ebenfalls in der «wir»-Form formuliert.
(A.207) Anna Felicitas Sarholz wollte nicht aufgeben. Jennifer Zietz hatte schon verschossen, Weltmeisterin Anja Mittag ebenfalls – **die Spielerinnen von Turbine Potsdam** ließen die Köpfe hängen während des Elfmeterschießens im Finale der Champions League **gegen Turbine Potsdam**.	Süddeutsche Zeitung, 22./23./24. Mai 2010, S. 40, «Coole Elfmeterheldin» (ffu)		Die genannten Spielerinnen gehören zum Verein Turbine Potsdam, jedoch spielten sie nicht gegen sich selbst, sondern gegen Olympique Lyon.
(A.208) Plasberg gibt sich gelassen, der Mann mit angeschlossener Produktionsfirma ist im ARD Programm und im WDR-Fernsehen breit aufgestellt – anders Anne Will: ihre Produktionsfirma steht für eine Sendung. **Ob die sie ARD verlässt.**	Der Tagesspiegel, 12. Juni 2010, S. 31, «Grosse Mutter ARD. Fünf Tage, sechs Köpfe: Mit Günther Jauch hat das Erste zu viel Personal – und zu wenig Lösungen» (Joachim Huber, Kurt Sagatz)	Satzstruktur	Vermutlich ist der nicht mehr verständliche Satz der Rest eines Satzes, aus dem verschiedene Elemente an verschiedene andere Stellen kopiert wurden.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.209) Die Wahl des Präsidenten ist am 1. Juni, zum ersten Mal wählt ihn der Hochschulrat, **dem zur Hälfte aus externe Persönlichkeiten angehören**.	Süddeutsche Zeitung, 20. April/1./2. Mai 2010, S. 42, «Studenten votieren für Nida-Rümelin» (math)	Fehlende Wörter, Wortstellung	Verschiedene Zwischenversionen des letzten Teilsatzes sind denkbar: «*der zur Hälfte aus externen Persönlichkeiten besteht*» oder «*dem zur Hälfte externe Persönlichkeiten angehören*».
(A.210) Vergangenes Jahr hatte frühere Finanz- und jetzige Innenminister Rainer Speer (SPD) noch Bilder eines überaus hellen, nach oben hin zur einer Glaskonstruktion offenen Saal präsentiert.	Uckermark Kurier, 6. Juli 2010, S. 5, «Streit schwelt um veränderten Plenarsaal» (Alexander Fröhlich)	Fehlende Wörter, Artikel, Wortstellung	Hier sind gleich mehrere Fehler enthalten: Es fehlt ein Artikel («*der frühere Finanz- …*»), die Substantivgruppe «*offenen Saal*» müsste im Genitiv stehen.
(A.211) George Washington heißt bei den Haudenosaunee seitdem Hanadahguyus, „**Stadzerstörerenn** sie einen Brief an den Präsidenten schreiben, dann lautet die Anrede stets: Dear Hanadahguyus!	Süddeutsche Zeitung, 16. Juli 2010, S. 16, «Das letzte Spiel der freien Irokesen» (Claus Biegert)	Interpunktion, Rechtschreibung	Die Anführungszeichen werden nicht mehr geschlossen, offenbar sollte damit die Übersetzung von Hanadahguyus markiert werden. Zudem ist das Ende dieser Übersetzung mit dem Beginn des nächsten Nebensatzes verschmolzen; korrekt wäre «*…«Stadtzerstörer», wenn sie einen …*». Im Original ist durch den Zeilenumbruch noch eine zusätzliche Trennung eingefügt: „Stadtzerstöre- renn sie einen
(A.212) Im Lorrainebad steigt das zweite «Säbeli-Bum»-Fest mit. Ein sommerliches Treffen von und mit Menschen mit und ohne Behinderung.	Blick am Abend (Ausgabe Bern), 19.8.2010, S. 7, «Prächtiger Jan für ein Prachts-Weekend» (PP)	Fehlende Wörter	Der erste Satz ist nicht vollständig.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.213) Das Frachtschiff mit rund 500 illegalen Immigranten, **darunter** denen Sicherheitskräfte auch Terroristen vermuten, befindet sich vor Vancouver Island an der Westküste Kanadas.	Süddeutsche Zeitung, 14./15. August 2010, S. 9, «Tiger an Bord. Ein Flüchtlingsschiff aus Sri Lanka alarmiert Kanada» (Bernadette Calonego)	Wortwahl	Der Fehler ist schwer zu entdecken, «*darunter*» ist getrennt, so dass «*dar-*» auf der einen und «*unter*» auf der anderen Zeile steht, dadurch erscheint die Variante «*unter denen Sicherheitskräfte auch Terroristen vermuten, …*», was korrekt wäre.
(A.214) «Eine Regierung darf nicht Öffentlichkeitsarbeit zu Parteizwecken betreiben», sagte Badura, der selbst auch **als Gutachten** zum Beispiel für das Umweltministerium arbeitet.	Süddeutsche Zeitung, 9. August 2010, S. 23, «Seehofer unter Druck. Der SPD-Chef im Landtag legt nach: «Staatskanzlei hat Öffentlichkeit belogen»» (Katja Auer)	Wortwahl	Entweder arbeitet Peter Badura als «*Gutachter*» oder er «*erstellt*» Gutachten für das Ministerium.
(A.215) **Ich** habe **ich** das SVEB1-Diplom gemacht und **ich** verfüge **ich** ausreichend Trainingspraxis, das ich auch auf höherem Level, aber auch in der Basisausbildung eingesetzt werden kann.	Aus einem Bewerbungsschreiben, September 2010	Doppeltes Auftreten von Wörtern	Das Subjektprononmen tritt jeweils doppelt auf. Vermutlich wurde der Satz erst später als eigenständiger Satz umformuliert und war vorher ein Teilsatz eines anderen, in dem das Pronomen nach dem Verb folgte.
(A.216) An der Ampel in der Sachsenhausener Straße, Höhe Rungestraße, Bernauer Straße, **Höhe Alkohol und Geld haben kollisierten** gestern um 18:43 drei Fahrzeuge.	Märkische Allgemeine Zeitung, 2. Juni 2010, «Polizeibericht»	Doppeltes Auftreten von Wörtern, Wortstellung, Wortwahl, Verben, Subtantivwortgruppen	Das Verb «*kollisieren*» ist offenbar von «*Kollision*» abgeleitet worden, existiert so jedoch nicht. Die Nennung von «*Alkohol und Geld*» ist völlig unverständlich. Das zweite Auftreten von «*Höhe*» ist entweder überflüssig oder es fehlt eine zusätzliche Angabe, etwa die Nennung eines markanten Gebäudes.

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.217) **Für** die Opposition **sehen** sich nun zusätzlich bestärkt, einen Untersuchungsausschuss einzuberufen und **fordert**, den Verkauf rückgängig zu machen.	Uckermark Kurier, 17. September, S. 5, «Opposition beharrt auf Untersuchung. Es gibt weiter verhärtete Fronten im Streit um Verkauf der Krampnitz-Kasernen» (FROE)	Kongruenz, überflüssige Wörter	Würde das erste Wort des Satzes («*Für*») entfernt, bliebe lediglich der Kongruenzfehler im ersten Teilsatz («*die Opposition sehen sich*»).
(A.218) Die strittige Szene: Diegos Fallrückzieher **geht** Levels wird zum Unfall – in der Folge trifft Kahlenberg (nicht im Bild) zum 1:0 für Wolfsburg.	Süddeutsche Zeitung, 4. Oktober 2010, S. 24, Bildunterschrift	Verben	Das Verb «*geht*» ist hier überflüssig. Streicht man es, ist der Satz jedoch weiterhin nicht verständlich.
(A.219) Erst vor Tagen hatte das kolumbianische Militär gemeldet, man habe **ein einem** Luftangriff elf Leute der Farc getötet, darunter auch einen ihrer Kommandanten.	Der Tagesspiegel, 23. Dezember 2009, S. 5, «Farc-Rebellen in Kolumbien entführen Gouverneur»	Präpositionen	Hier hat vermutlich die automatische Rechtschreibkorrektur einen Teil zum Fehler beigetragen. «*ein*» könnte «*bei*» oder «*in*» heissen.
(A.220) Leila und ich waren im Sommer schon vier Wochen zu Besuch bei einer Freundin. Ich bin **mit nachts mit** dem Auto durch die Straßen gefahren, habe Radio gehört und hatte seit langem wieder das Gefühl, dass ich lächeln kann.	Tagesspiegel, 14. November 2010, S. S7, ««Ich habe mich so sehr gehasst». Interview mit Nadja Benaissa» (Esther Kogelboom, Ulf Lippitz)	Fehlende Wörter, überflüssige Wörter	Hier ist entweder das erste «mit» zuviel und es müsste heissen «Ich bin nachts mit dem Auto durch die Straßen gefahren» oder es fehlt das Objekt nach der Präposition und könnte heissen «Ich bin mit ihr nachts mit dem Auto».

Fehlertext	Quelle	Kategorie(n)	Kommentar
(A.221) Kürzestmögliche Sequenz **in von** Operationen MS *Word*.	Rohfassung der vorliegenden Dissertation, Cerstin Mahlow	Wortstellung	Hier wurde «*von Operationen*» nachträglich in den Satz «*Kürzestmögliche Sequenz in* MS Word» eingefügt und der Ort der Einfügung falsch gewählt.

Literaturverzeichnis

[Abney 1991] Steven Abney. Parsing by Chunks. In Robert Berwick, Steven Abney und Carol Tenny (Hg.) *Principle-Based Parsing*. Boston, Dordrecht, London: Kluwer, 1991. (Zitiert auf S. 167 und 172)

[Alamargot und Chanquoy 2001] Denis Alamargot und Lucile Chanquoy. General introduction. In Denis Alamargot und Lucile Chanquoy (Hg.) *Through the Models of Writing*, Bd. 9 von *Studies in Writing*, 1–29. Boston, Dordrecht, London: Kluwer, 2001. (Zitiert auf S. 1, 21, 22 und 25)

[Alamargot et al. 2006] Denis Alamargot, David Chesnet, Christophe Dansac und Christine Ros. Eye and Pen: A new device for studying reading during writing. In *Behavior Research Methods*, 38(2):287–299, 2006. (Zitiert auf S. 116)

[Allal und Chanquoy 2004] Linda Allal und Lucile Chanquoy. Introduction: Revision Revisited. In Linda Allal, Lucile Chanquoy und Pierre Largy (Hg.) *Revision. Cognitive and instructional processes*, Bd. 13 von *Studies in Writing*, 1–7. Boston, Dordrecht, London: Kluwer, 2004. (Zitiert auf S. 38 und 39)

[Allen und Scerbo 1983] Robert B. Allen und Mark W. Scerbo. Details of command-language keystrokes. In *ACM Transactions on Information Systems*, 1(2):159–178, 1983. doi: 10.1145/357431.357434. (Zitiert auf S. 24, 80 und 96)

[Allen et al. 1981] Todd Allen, Robert Nix und Alan Perlis. PEN: A hierarchical document editor. In *Proceedings of the ACM SIGPLAN SIGOA symposium on Text manipulation*, 74–81. New York, NY, USA: ACM, 1981. doi: 10.1145/800209.806457. (Zitiert auf S. 50 und 80)

[Andriessen et al. 1996] Jerry Andriessen, Koenraad De Smedt und Michael Zock. Discourse planning: Empirical research and computer models. In Anton Dijkstra und Koenraad De Smedt (Hg.) *Computational psycholinguistics: AI and connectionist models of human language processing*, 247–278. London, GB: Taylor & Francis, 1996. (Zitiert auf S. 22, 24, 32 und 78)

[Babaian et al. 2002] Tamara Babaian, Barbara J. Grosz und Stuart M. Shieber. A writer's collaborative assistant. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, 7–14. New York, NY, USA: ACM, 2002. doi: 10.1145/502716.502722. (Zitiert auf S. 73 und 241)

[Ballance et al. 1992] Robert A. Ballance, Susan L. Graham und Michael L. Van De Vanter. The Pan language-based editing system. In *ACM Transactions on Software Engineering Methodology*, 1(1):95–127, 1992. doi: 10.1145/125489.122804. (Zitiert auf S. 7, 80 und 96)

[Beesley und Karttunen 2003] Kenneth R. Beesley und Lauri Karttunen. *Finite State Morphology*. Stanford, CA, USA: Center for the Study of Language and Information, 2003. (Zitiert auf S. 132)

[Bereiter und Scardamalia 1987] Carl Bereiter und Marlene Scardamalia. *The psychology of written composition*. Hillsdale, NJ, USA: Lawrence Erlbaum, 1987. (Zitiert auf S. xv, 20 und 21)

[Bergin 2006a] Thomas J. Bergin. The Origins of Word Processing Software for Personal Computers: 1976–1985. In *IEEE Annals of the History of Computing*, 28(4):32–47, 2006. doi: 10.1109/MAHC.2006.76. (Zitiert auf S. 48, 49, 50, 106 und 107)

[Bergin 2006b] Thomas J. Bergin. The Proliferation and Consolidation of Word Processing Software: 1985–1995. In *IEEE Annals of the History of Computing*, 28(4):48–63, 2006. doi: 10.1109/MAHC.2006.77. (Zitiert auf S. 49, 50 und 106)

[Blandy 2009] Jim Blandy. GNU Emacs: Creeping featurism is a strength. In Diomidis Spinellis und Georgios Gousios (Hg.) *Beautiful Architecture*, Kap. 11, 263–277. Sebastopol, CA, USA: O'Reilly, 2009. (Zitiert auf S. 188, 192 und 197)

[Blatt 2004] Inge Blatt. Schreiben und Schreibenlernen mit neuen Medien. Eine Bestandsaufnahme. In Inge Blatt und Wilfried Hartmann (Hg.) *Schreibprozesse im medialen Wandel. Ein Studienbuch*, 30–70. Baltmannsweiler, Deutschland: Schneider Verlag Hohengehren, 2004. (Zitiert auf S. 29, 38 und 69)

[Boehm 1993] Diane C. Boehm. Mozartians, Beethovians, and the teaching of writing. In *Quarterly of the National Writing Project and the Center for the Study of Writing and Literacy*, 15(2):15–18, 1993. (Zitiert auf S. 110)

[Bolter 1989] Jay D. Bolter. Beyond word processing: The computer as a new writing space. In *Language & Communication*, 9(2–3):129–142, 1989. doi: 10.1016/0271-5309(89)90014-1. (Zitiert auf S. 42 und 56)

[van den Bosch und Buchholz 2002] Antal van den Bosch und Sabine Buchholz. Shallow parsing on the basis of words only: A case study. In *ACL '02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL '02, 433–440. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002. doi: 10.3115/1073083.1073156. (Zitiert auf S. 164)

[Boyd und Meurers 2008] Adriane Boyd und Detmar Meurers. Revisiting the impact of different annotation schemes on PCFG parsing: A grammatical

dependency evaluation. In *PaGe '08: Proceedings of the Workshop on Parsing German*, 24–32. Morristown, NJ, USA: Association for Computational Linguistics, 2008. (Zitiert auf S. 165)

[Brants et al. 2002] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius und George Smith. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, 24–41. 2002. (Zitiert auf S. 164)

[Brants et al. 1997] Thorsten Brants, Roland Hendriks, Sabine Kramp, Brigitte Krenn, Cordula Preis, Wojciech Skut und Hans Uszkoreit. Das NEGRA-Annotationsschema. Techn. Ber., Universität des Saarlandes, Saarbrücken, Deutschland, 1997. (Zitiert auf S. 164)

[Breidenbach 2006] Cathleen Breidenbach. Practical Guidelines for Writers and Teachers. In Alice Horning und Anne Becker (Hg.) *Revision: History, Theory, and Practice*, Reference Guides to Rhetoric and Composition, Kap. 11, 197–219. West Lafayette, IN, USA: Parlor Press, 2006. (Zitiert auf S. 30 und 88)

[Bridwell 1980] Lillian S. Bridwell. Revising Strategies in Twelfth Grade Students' Transactional Writing. In *Research in the Teaching of English*, 14(3):197–222, 1980. (Zitiert auf S. 38, 89, 92 und 110)

[Buck 2008] Amber M. Buck. The invisible interface: MS Word in the writing center. In *Computers and Composition*, 25(4):396–415, 2008. doi: 10.1016/j.compcom.2008.05.003. (Zitiert auf S. 117)

[Callender 1982] E. David Callender. An evaluation of the AUGMENT system. In *SIGDOC '82: Proceedings of the 1st annual international conference on Systems documentation*, 29–35. New York, NY, USA: ACM Press, 1982. doi: 10.1145/800065.801306. (Zitiert auf S. 49 und 107)

[Calonne 2006] David S. Calonne. Creative Writers and Revision. In Alice Horning und Anne Becker (Hg.) *Revision: History, Theory, and Practice*, Reference Guides to Rhetoric and Composition, Kap. 9, 142–176. West Lafayette, IN, USA: Parlor Press, 2006. (Zitiert auf S. 110)

[Carlson 1990] Patricia A. Carlson. Artificial neural networks as cognitive tools for professional writing. In *SIGDOC '90: Proceedings of the 8th annual international conference on Systems documentation*, 95–110. New York, NY, USA: ACM, 1990. doi: 10.1145/97426.97997. (Zitiert auf S. 107)

[Carr 2010] Nicholas Carr. *The Shallows: What the Internet Is Doing to Our Brains*. New York, NY, USA: W. W. Norton & Company, 2010. (Zitiert auf S. 112)

[Chamberlin et al. 1981] Donald D. Chamberlin, James C. King, Donald R. Slutz, Stephen J. P. Todd und Bradford W. Wade. JANUS: An interactive system for document composition. In *Proceedings of the ACM SIGPLAN SIGOA symposium on Text manipulation*, 82–91. New York, NY, USA: ACM, 1981. doi: 10.1145/800209.806458. (Zitiert auf S. 80)

[Chanquoy et al. 1990] Lucile Chanquoy, Jean-Noël Foulin und Michel Fayol. Temporal management of short text writing by children and adults. In *Cahiers de psychologie cognitive*, 10(5):513–540, 1990. (Zitiert auf S. 30)

[Chenoweth und Hayes 2003] N. Ann Chenoweth und John R. Hayes. The Inner Voice in Writing. In *Written Communication*, 20(1):99–118, 2003. doi: 10.1177/0741088303253572. (Zitiert auf S. 23, 51 und 109)

[Cherry 1981] Lorinda Cherry. Computer aids for writers. In *ACM SIGOA Newsletter*, 2(1-2):61–67, 1981. doi: 10.1145/1159890.806455. (Zitiert auf S. 1)

[Chomsky 1965] Noam Chomsky. *Aspects of the Theory of Syntax*. Cambridge, MA, USA: MIT Press, 1965. (Zitiert auf S. 11)

[Church 1988] Kenneth W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, 136–143. Morristown, NJ, USA: Association for Computational Linguistics, 1988. doi: 10.3115/974235.974260. (Zitiert auf S. 167 und 172)

[Chusho et al. 1983] Takeshi Chusho, Tan Watanabe und Toshihiro Hayashi. A Language-Adaptive Programming Environment Based on a Program Analyzer and a Structure Editor. In R. E. A. Mason (Hg.) *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19–23, 1983*, 621–626. 1983. (Zitiert auf S. 80, 85 und 86)

[Clematide 2008] Simon Clematide. An OLIF-based open inflectional resource and yet another morphological system for German. In Angelika Storrer, Alexander Geyken, Alexander Siebert und Kay-Michael Würzner (Hg.) *Text Resources and Lexical Knowledge: Selected Papers from the 9th Conference on Natural Language Processing KONVENS 2008*, 183–194. Berlin, Deutschland: Mouton de Gruyter, 2008. (Zitiert auf S. 126 und 139)

[Collier 1983] Richard M. Collier. The Word Processor and Revision Strategies. In *College Composition and Communication*, 34(2):149–155, 1983. doi: 10.2307/357402. (Zitiert auf S. 37)

[Collier und Werier 1995] Richard M. Collier und Clifford Werier. When computer writers compose by hand. In *Computers and Composition*, 12(1):47–59, 1995. doi: 10.1016/8755-4615(95)90022-5. (Zitiert auf S. 18, 24, 102, 103 und 116)

[Cookson 1989] Stefanie Cookson. Designing Computational Writing Tools within a Linguistic Model of the Writing Process. In Noel Williams und Patrik O'Brian Holt (Hg.) *Computers and Writing: Models and tools*, Kap. 2, 17–21. Oxford, GB: Intellect, 1989. (Zitiert auf S. 243)

[Cooper et al. 2007] Alan Cooper, Robert Reimann und David Cronin. *About Face 3: The Essentials of Interaction Design*. Indianapolis, IN, USA: Wiley, 3rd Aufl., 2007. (Zitiert auf S. 95 und 123)

[Daelemans und van den Bosch 2005] Walter Daelemans und Antal van den Bosch. *Memory-Based Language Processing*. Studies in Natural Language Processing. Cambridge, GB: Cambridge University Press, 2005. (Zitiert auf S. 164)

[Daelemans et al. 1996] Walter Daelemans, Jakub Zavrel, Peter Berck und Steven Gillis. MBT: A memory-based part of speech tagger-generator. In Eva Ejerhed und Ido Dagan (Hg.) *Proceedings of the Fourth Workshop on Very Large Corpora*, 14–27. 1996. (Zitiert auf S. 163 und 200)

[Daelemans et al. 2010] Walter Daelemans, Jakub Zavrel, Antal van den Bosch und Ko van der Sloot. MBT: Memory-Based Tagger version 3.2 reference guide. Techn. Ber., Induction of Linguistic Knowledge Research Group, Department of Communication and Information Sciences, Tilburg University, Tilburg, Niederlande, 2010. (Zitiert auf S. 163 und 200)

[Daiute 1983] Colette A. Daiute. The Computer as Stylus and Audience. In *College Composition and Communication*, 34(2):134–145, 1983. doi: 10.2307/357400. (Zitiert auf S. 1, 33, 36 und 102)

[Daiute und Taylor 1981] Colette A. Daiute und Robert Taylor. Computers and the improvement of writing. In *ACM 81: Proceedings of the ACM '81 conference*, 83–88. New York, NY, USA: ACM, 1981. doi: 10.1145/800175.809841. (Zitiert auf S. 7, 33 und 42)

[Dale 1989] Robert Dale. Computer-based Editorial Aids. In Jeremy Peckham (Hg.) *Recent Developments and Applications of Natural Language Processing*, Kap. 2, 8–22. London, GB: Kogan Page, 1989. (Zitiert auf S. 61, 62 und 67)

[Dale 1990] Robert Dale. A Rule-Based Approach to Computer-Assisted Copy-Editing. In *Computer Assisted Language Learning*, 2(1):59–67, 1990. (Zitiert auf S. 38, 61 und 62)

[Dale 1997] Robert Dale. Computer assistance in text creation and editing. In Giovanni B. Varile, Antonio Zamponelli, Ronald Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen und Victor Zue (Hg.) *Survey of the State of the Art in Human Language Technology*, Studies in Natural Language Processing, 235–237. Cambridge, New York, Melbourne: Cambridge University Press, 1997. (Zitiert auf S. 67, 84 und 243)

[Dale und Douglas 1996] Robert Dale und Shona Douglas. Two Investigations into Intelligent Text Processing. In Mike Sharples und Thea van der Geest (Hg.) *The New Writing Environment: Writers at Work in a World of Technology*, Kap. 8, 123–145. Berlin, Heidelberg, New York: Springer, 1996. (Zitiert auf S. 59, 61, 62, 64 und 67)

[van Dam und Rice 1971] Andries van Dam und David E. Rice. On-line Text Editing: A Survey. In *ACM Computing Surveys*, 3(3):93–114, 1971. doi: 10.1145/356589.356591. (Zitiert auf S. 5, 49 und 50)

[Day 1988] John T. Day. Writer's Workbench: A useful aid, but not a cure-all. In *Computers and Composition*, 6(1):63–78, 1988. doi: 10.1016/S8755-4615(88)80028-8. (Zitiert auf S. 58 und 70)

[De Smedt 2009] Koenraad De Smedt. NLP for writing: What has changed? In Rickard Domeij, Sofie Johansson Kokkinakis, Ola Knutsson und Sylvana Sofkova Hashemi (Hg.) *Proceedings of the Workshop on NLP for Reading and Writing – Resources, Algorithms and Tools*, 1–11. Northern European Association for Language Technology (NEALT), 2009. (Zitiert auf S. 85)

[De Smedt und Kempen 1987] Koenraad De Smedt und Gerard Kempen. Incremental Sentence Production, Self-Correction and Coordination. In Gerard Kempen (Hg.) *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, Kap. 23, 365–376. Dordrecht, Boston, Lancaster: Martinus Nijhoff, 1987. (Zitiert auf S. 83)

[Dijkstra 1972] Edsger W. Dijkstra. Notes on structured programming. In *Structured Programming*, Kap. 1, 1–82. London, GB: Academic Press, 1972. (Zitiert auf S. 80)

[DiPardo 1994] Anne DiPardo. Stimulated recall in research on writing: An antidote to "I don't know, it was fine.". In Peter Smagorinsky (Hg.) *Speaking about writing: Reflections on research methodology*, Series on Written Communication, 163–181. Thousand Oaks, CA, USA: Sage Publications, 1994. (Zitiert auf S. 42)

[Donalies 1999] Elke Donalies. Präfixverben, Halbpräfixverben, Partikelverben, Konstitutionsverben oder verbale Gefüge? Ein Analyseproblem der deutschen Wortbildung. In *Studia Germanica Universitatis Vesprimiensis*, 3(2):127–143, 1999. (Zitiert auf S. 129)

[Donalies 2005] Elke Donalies. *Die Wortbildung des Deutschen*. Tübingen, Deutschland: Gunter Narr, 2005. (Zitiert auf S. 128 und 129)

[Dorner 1992] Jane Dorner. Authors and Information Technology: New Challenges in Publishing. In Mike Sharples (Hg.) *Computers and Writing: Issues and Implementation*, 5–14. Boston, Dordrecht, London: Kluwer, 1992. (Zitiert auf S. 68, 102 und 105)

[Dowling 1994] Carolyn Dowling. Word processing and the ongoing difficulty of writing. In *Computers and Composition*, 11(3):227–235, 1994. doi: 10.1016/8755-4615(94)90015-9. (Zitiert auf S. 25, 35 und 103)

[Duffy und Robinson 1996] Thomas M. Duffy und Carol A. Robinson. Designing Tools to Aid Technical Editors: A Needs Analysis. Techn. Ber., Navy Personnel Research and Development Center, San Diego, CA, USA, 1996. (Zitiert auf S. 68, 69 und 206)

[Eberwein 2005] Dieter Eberwein. *Nietzsches Schreibkugel. Ein Blick auf Nietzsches Schreibmaschinenzeit durch die Restauration der Schreibkugel*. Schauenburg, Deutschland: Typoskript, 2005. (Zitiert auf S. 112 und 113)

[Edwards 2005] Jonathan Edwards. Subtext: Uncovering the simplicity of programming. In *OOPSLA '05: Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications*, 505–518. New York, NY, USA: ACM, 2005. doi: 10.1145/1094811.1094851. (Zitiert auf S. 81)

[Eisenberg 1992] Daniel Eisenberg. History of Word Processing. In *Encyclopedia of Library and Information Science*, (49):268–278, 1992. (Zitiert auf S. 6, 48, 49, 50, 103 und 106)

[Eisenberg 2006] Peter Eisenberg. *Grundriss der deutschen Grammatik 1: Das Wort*. Stuttgart, Deutschland: J.B. Metzler, 2006. (Zitiert auf S. 128)

[Embley und Nagy 1981] David W. Embley und George Nagy. Behavioral Aspects of Text Editors. In *ACM Computing Surveys*, 13(1):33–70, 1981. doi: 10.1145/356835.356838. (Zitiert auf S. 83)

[Engelbart 1962] Douglas C. Engelbart. AUGMENTING HUMAN INTELLECT: A Conceptual Framework. Techn. Ber., Stanford Research Institute, Stanford, CA, USA, 1962. (Zitiert auf S. 49 und 107)

[Ericsson und Simon 1980] K. Anders Ericsson und Herbert A. Simon. Verbal reports as data. In *Psychological Review*, 87:215–251, 1980. (Zitiert auf S. 23)

[Evert 2004] Stefan Evert. The Statistical Analysis of Morphosyntactic Distributions. In *LREC 2004 Fourth International Conference on Language Resources and Evaluation*, 1539–1542. Paris, Frankreich: ELRA, 2004. (Zitiert auf S. 124, 165, 168, 175, 176 und 179)

[Evert 2005] Stefan Evert. *The statistics of word cooccurrences: Word pairs and collocations*. Dissertation, Universität Stuttgart, Institut für Maschinelle Sprachverarbeitung, Stuttgart, Deutschland, 2005. (Zitiert auf S. 10)

[Eyman und Reilly 2006] Douglas Eyman und Colleen Reilly. Revising with Word Processing/Technology/Document Design. In Alice Horning und Anne Becker (Hg.) *Revision: History, Theory, and Practice*, Reference Guides to Rhetoric and Composition, Kap. 7, 102–116. West Lafayette, IN, USA: Parlor Press, 2006. (Zitiert auf S. 36, 37 und 69)

[Faigley und Witte 1981] Lester Faigley und Stephen Witte. Analyzing Revision. In *College Composition and Communication*, 32(4):400–414, 1981. doi: 10.2307/356602. (Zitiert auf S. 23, 30, 39, 40, 89 und 92)

[Fayol 1999] Michel Fayol. From on-line management problems to strategies in written composition. In Mark Torrance und Gaynor Jeffery (Hg.) *The Cognitive Demands of Writing: Processing Capacity and Working Memory Effects in Text Production*, Bd. 3 von *Studies in Writing*, 13–23. Amsterdam, Niederlande: Amsterdam University Press, 1999. (Zitiert auf S. 1, 24 und 25)

[Ferret et al. 1998] Olivier Ferret, Brigitte Grau und Nicolas Masson. Thematic segmentation of texts: Two methods for two kinds of texts. In *Proceedings of the 17th international conference on Computational linguistics*, 392–396. Morristown, NJ, USA: Association for Computational Linguistics, 1998. doi: 10.3115/980845.980912. (Zitiert auf S. 162)

[Fitzgerald 1987] Jill Fitzgerald. Research on Revision in Writing. In *Review of Educational Research*, 57(4):481–506, 1987. doi: 10.2307/1170433. (Zitiert auf S. 29, 30 und 39)

[Flinn 1987] Jane Z. Flinn. Case studies of revision aided by keystroke recording and replaying software. In *Computers and Composition*, 5(1):31–44, 1987. doi: 10.1016/S8755-4615(87)80013-0. (Zitiert auf S. 32 und 43)

[Flower und Hayes 1981] Linda S. Flower und John R. Hayes. A Cognitive Process Theory of Writing. In *College Composition and Communication*, 32(4):365–387, 1981. doi: 10.2307/356600. (Zitiert auf S. xv, 19, 22, 24 und 61)

[Flower et al. 1986] Linda S. Flower, John R. Hayes, Linda Carey, Karen Schriver und James Stratman. Detection, Diagnosis, and the Strategies of Revision. In *College Composition and Communication*, 37(1):16–55, 1986. (Zitiert auf S. 30)

[Fontenelle 2005] Thierry Fontenelle. Dictionnaires et outils de correction linguistique. In *Revue française de linguistique appliquée*, X(2):119–128, 2005. (Zitiert auf S. 64 und 68)

[Foster 2010] Jennifer Foster. "cba to check the spelling": Investigating Parser Performance on Discussion Forum Posts. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 381–384. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (Zitiert auf S. 165)

[Fraser 1981] Christopher W. Fraser. Syntax-directed editing of general data structures. In *SIGPLAN Notices*, 16(6):17–21, 1981. doi: 10.1145/872730.806449. (Zitiert auf S. 80)

[Freed und Borenstein 1996] Edwin E. Freed und Nathaniel S. Borenstein. Multipurpose Internet Mail Extensions (MIME). Part Two: Media Types. Techn. Ber. RFC 2046, Internet Engineering Task Force, 1996. URL http://www.rfc-editor.org/rfc/rfc2046.txt. (Zitiert auf S. 5 und 6)

[Frisch 2010] Max Frisch. *Entwürfe zu einem dritten Tagebuch*. Berlin, Deutschland: Suhrkamp, 2010. (Zitiert auf S. 28 und 31)

[Galbraith und Torrance 2004] David Galbraith und Mark Torrance. Revision in the Context of Different Drafting Strategies. In Linda Allal, Lucile Chanquoy und Pierre Largy (Hg.) *Revision. Cognitive and instructional processes*, Bd. 13 von *Studies in Writing*, 63–85. Boston, Dordrecht, London: Kluwer, 2004. (Zitiert auf S. 96)

[Gamon 2010] Michael Gamon. Using Mostly Native Data to Correct Errors in Learners' Writing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 163–171. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (Zitiert auf S. 71)

[van der Geest 1996] Thea van der Geest. Studying "real life" writing processes: A proposal and an example. In C. Michael Levy und Sarah Ransdell (Hg.) *The Science of Writing. Theories, Methods, Individual Differences, and Applications*, 309–322. Hillsdale, NJ, USA: Lawrence Erlbaum, 1996. (Zitiert auf S. 117)

[Geisler et al. 2001] Cheryl Geisler, Charles Bazerman, Stephen Doheny-Farina, Laura Gurak, Christina Haas, Johndan Johnson-Eilola, David S. Kaufer, Andrea Lunsford, Carolyn R. Miller, Dorothy Winsor und Joanne Yates. IText: Future Directions for Research on the Relationship between Information Technology and Writing. In *Journal of Business and Technical Communication*, 15(3):269–308, 2001. (Zitiert auf S. 108)

[Geyken 2009] Alexander Geyken. Automatische Wortschatzerschließung großer Textkorpora am Beispiel des DWDS. In *Linguistik online*, (39):97–107, 2009. URL http://www.linguistik-online.de/39_09/geyken.html. (Zitiert auf S. 126)

[Geyken und Hanneforth 2006] Alexander Geyken und Thomas Hanneforth. TAGH: A complete morphology for German based on weighted finite state automata. In Anssi Yli-Jyrä, Lauri Karttunen und Juhani Karhumäki (Hg.) *Finite-State Methods and Natural Language Processing*, Bd. 4002 von *Lecture Notes in Computer Science*, Kap. 7, 55–66. Berlin, Heidelberg, New York: Springer, 2006. doi: 10.1007/11780885_7. (Zitiert auf S. 126)

[Good 1981] Michael Good. Etude and the folklore of user interface design. In *Proceedings of the ACM SIGPLAN SIGOA symposium on Text manipulation*, 34–43. New York, NY, USA: ACM, 1981. doi: 10.1145/800209.806452. (Zitiert auf S. 95 und 124)

[Good 1985] Michael Good. The use of logging data in the design of a new text editor. In *SIGCHI Bulletin*, 16(4):93–97, 1985. doi: 10.1145/1165385.317474. (Zitiert auf S. 44)

[Gouge et al. 1983] Véronique D. Gouge, Gilles Kahn, Bernard Lang, Bertrand Melese und Elham Morcos. Outline of a Tool for Document Manipulation. In R. E. A. Mason (Hg.) *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19–23, 1983*, 615–620. 1983. (Zitiert auf S. 80)

[Grabowski 2008] Joachim Grabowski. The internal structure of university students' keyboard skills. In *Journal of Writing Research*, 1(1):27–52, 2008. (Zitiert auf S. 103, 104 und 115)

[Gubelmann 2010] Reto Gubelmann. *Systematischer Überblick über die Fundstücke im Fehlerforum zu "Making word Processors process Words"*. Seminararbeit, Universität Zürich, Insitut für Computerlinguistik, Zürich, Schweiz, 2010. (Zitiert auf S. 4, 66 und 246)

[Gustafson-Čapková et al. 2007] Sofia Gustafson-Čapková, Yvonne Samuelsson und Martin Volk. SMULTRON (version 1.0) – the Stockholm MULtilingual parallel TReebank. Online, 2007. URL `http://www.ling.su.se/dali/research/smultron/index.htm`. (Zitiert auf S. 165)

[Haapalainen und Majorin 1994] Mariikka Haapalainen und Ari Majorin. GERTWOL: Ein System zur automatischen Wortformerkennung deutscher Wörter. Techn. Ber., Lingsoft, Inc., 1994. (Zitiert auf S. 126, 197 und 231)

[Haigh 2006] Thomas Haigh. Remembering the Office of the Future: The Origins of Word Processing and Office Automation. In *IEEE Annals of the History of Computing*, 28(4):6–31, 2006. doi: 10.1109/MAHC.2006.70. (Zitiert auf S. 6, 47, 48, 49 und 50)

[Hall und Nivre 2008] Johan Hall und Joakim Nivre. A dependency-driven parser for German dependency and constituency representations. In *PaGe '08: Proceedings of the Workshop on Parsing German*, 47–54. Morristown, NJ, USA: Association for Computational Linguistics, 2008. (Zitiert auf S. 165)

[Hamlet 1986] Richard Hamlet. A Disciplined Text Environment. In J. C. van Vliet (Hg.) *Text Processing and Document Manipulation: Proceedings of the International Conference*, 78–89. Cambridge, GB: Cambridge University Press, 1986. (Zitiert auf S. 4, 5, 82, 94 und 218)

[Handl et al. 2009] Johannes Handl, Besim Kabashi, Thomas Proisl und Carsten Weber. JSLIM – Computational morphology in the framework of the SLIM theory of language. In Cerstin Mahlow und Michael Piotrowski (Hg.) *State of the Art in Computational Morphology*, Bd. 41 von *Communications in Computer and Information Science*, Kap. 2, 10–27. Berlin, Heidelberg, New York: Springer, 2009. doi: 10.1007/978-3-642-04131-0_2. (Zitiert auf S. 126)

[Hanrieder 1996] Gerhard Hanrieder. MORPH – Ein modulares und robustes Morphologieprogramm für das Deutsche in Common Lisp. In Roland Hausser

(Hg.) *Linguistische Verifikation. Dokumentation zur Ersten Morpholympics 1994*, Kap. 7, 53–66. Tübingen, Deutschland: Niemeyer, 1996. (Zitiert auf S. 126)

[Hansen 1971] Wilfred J. Hansen. User engineering principles for interactive systems. In *AFIPS '71 (Fall): Proceedings of the November 16-18, 1971, Fall Joint Computer Conference*, 523–532. New York, NY, USA: ACM, 1971. doi: 10.1145/1479064.1479159. (Zitiert auf S. 79 und 80)

[Hara et al. 2009] Kazuo Hara, Masashi Shimbo, Hideharu Okuma und Yuji Matsumoto. Coordinate Structure Analysis with Global Structural Constraints and Alignment-Based Local Features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 967–975. Morristown, NJ, USA: Association for Computational Linguistics, 2009. (Zitiert auf S. 213)

[Hardmeier et al. 2010] Christian Hardmeier, Arianna Bisazza und Marcello Federico. FBK at WMT 2010: Word lattices for morphological reduction and chunk-based reordering. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, 88–92. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (Zitiert auf S. 161)

[Hartley 2007] James Hartley. Longitudinal studies of the effects of new technologies on writing: Two case studies. In Mark Torrance, Luuk Van Waes und David Galbraith (Hg.) *Writing and Cognition: Research and Applications*, Bd. 20 von *Studies in Writing*, Kap. 19, 293–305. Amsterdam, Niederlande: Elsevier Science, 2007. (Zitiert auf S. 18)

[Hausdorf 2005] Carsten Hausdorf. *Integratives, ganzheitliches Modell und Werkzeug zur wissenschaftlichen Textproduktion*. Dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Deutschland, 2005. (Zitiert auf S. 6, 51, 107, 190 und 242)

[Hausser 1996] Roland Hausser. *Linguistische Verifikation. Dokumentation zur Ersten Morpholympics*. Tübingen, Deutschland: Niemeyer, 1996. (Zitiert auf S. 10, 126, 132 und 147)

[Hausser 2001] Roland Hausser. *Foundations of Computational Linguistics: Human-Computer Communication in Natural Language*. Berlin, Heidelberg, New York: Springer, 2nd rev. and ext. Aufl., 2001. (Zitiert auf S. 55, 127, 130 und 133)

[Hawisher 1986] Gail E. Hawisher. Studies in word processing. In *Computers and Composition*, 4(1):6–31, 1986. doi: 10.1016/S8755-4615(86)80003-2. (Zitiert auf S. 2, 33, 34 und 108)

[Hawisher 1988] Gail E. Hawisher. Research update: Writing and word processing. In *Computers and Composition*, 5(2):7–27, 1988. doi: 10.1016/8755-4615(88)80002-1. (Zitiert auf S. 33, 34 und 108)

[Hawisher et al. 1995] Gail E. Hawisher, Paul LeBlanc, Charles Moran und Cynthia L. Selfe. *Computers and the Teaching of Writing in American Higher Education, 1979–1994: A History*. New Directions in Computers and Composition Studies. New York, NY, USA: Ablex, 1995. (Zitiert auf S. 18 und 108)

[Hayes 1996] John R. Hayes. A new framework for understanding cognition and affect in writing. In C. Michael Levy und Sarah Ransdell (Hg.) *The Science of Writing. Theories, Methods, Individual Differences and Applications*, 1–27. Hillsdale, NJ, USA: Lawrence Erlbaum, 1996. (Zitiert auf S. xv, 20 und 21)

[Hayes 2001] John R. Hayes. Commentary on the book: Through the models of writing. In Denis Alamargot und Lucile Chanquoy (Hg.) *Through the Models of Writing*, Bd. 9 von *Studies in Writing*, 229–236. Boston, Dordrecht, London: Kluwer, 2001. (Zitiert auf S. 22, 61, 116 und 117)

[Hayes 2004] John R. Hayes. What triggers revision? In Linda Allal, Lucile Chanquoy und Pierre Largy (Hg.) *Revision. Cognitive and instructional processes*, Bd. 13 von *Studies in Writing*, 9–20. Boston, Dordrecht, London: Kluwer, 2004. (Zitiert auf S. 29)

[Hayes und Chenoweth 2006] John R. Hayes und N. Ann Chenoweth. Is Working Memory Involved in the Transcribing and Editing of Texts? In *Written Communication*, 23(2):135–149, 2006. doi: 10.1177/0741088306286283. (Zitiert auf S. 30)

[Heid 2010] Ulrich Heid. Computerlinguistik zwischen Informationswissenschaft und multilingualer Kommunikation. In *Information – Wissenschaft und Praxis*, 61(6–7):361–366, 2010. (Zitiert auf S. 9)

[Heidorn 2000] George E. Heidorn. Intelligent Writing Assistance. In Robert Dale, Herman Moisl und Harold Somers (Hg.) *Handbook of Natural Language Processing*, 181–207. New York, NY, USA: Marcel Dekker, 2000. (Zitiert auf S. 68)

[Heilker 1992] Paul Heilker. Revision Worship and the Computer as Audience. In *Computers and Composition*, 9(3):59–69, 1992. (Zitiert auf S. 68)

[Hernandez und Grau 2003] Nicolas Hernandez und Brigitte Grau. What is this text about? In *SIGDOC '03: Proceedings of the 21st annual international conference on Documentation*, 117–124. New York, NY, USA: ACM, 2003. doi: 10.1145/944868.944894. (Zitiert auf S. 162)

[Hess 1992] Michael Hess. Natural and formal language processing. In Gérard Comyn, Norbert E. Fuchs und Michael J. Ratcliffe (Hg.) *Logic Programming in Action*, Bd. 636 von *Lecture Notes in Computer Science*, 127–175. Berlin, Heidelberg, New York: Springer, 1992. doi: 10.1007/3-540-55930-2_12. (Zitiert auf S. 82 und 83)

[Hill et al. 1991] Charles A. Hill, David L. Wallace und Christina Haas. Revising on-line: Computer technologies and the revising process. In *Computers and Composition*, 9(1):83–109, 1991. doi: 10.1016/8755-4615(91)80040-K. (Zitiert auf S. 18, 24, 32 und 40)

[Hinrichs 2005] Erhard W. Hinrichs. Finite-state parsing of German. In Antti Arppe, Lauri Carlson, Krister Lindén, Jussi Piitulainen, Mickael Suominen, Martti Vainio, Hanna Westerlund und Anssi Yli-Jyrä (Hg.) *Inquiries into Words, Constraints, and Contexts. Festschrift for Kimmo Koskenniemi on his 60th Birthday*, CSLI Studies in Computational Linguistics, 35–44. Stanford, CA, USA: CSLI Publications, 2005. (Zitiert auf S. 167)

[Hirst 2008] Graeme Hirst. An evaluation of the contextual spelling checker of Microsoft Office Word 2007. Online, 2008. URL http://ftp.cs.toronto.edu/pub/gh/Hirst-2008-Word.pdf. (Zitiert auf S. 66 und 67)

[Hjelm und Schwarz 2005] Hans Hjelm und Christoph Schwarz. LiSa – morphological analysis for information retrieval. In Stefan Werner (Hg.) *Proceedings of the 15th NODALIDA conference, Joensuu 2005*, Bd. 1 von *Ling@JoY*, 65–70. 2005. (Zitiert auf S. 161)

[Holdstein und Redman 1985] Deborah H. Holdstein und Tim Redman. Empirical Research in Word-Processing: Expectations vs. Experience. In *Computers and Composition*, 3(1):43–54, 1985. (Zitiert auf S. 36)

[Hollink et al. 2004] Vera Hollink, Jaap Kamps, Christof Monz und Maarten De Rijke. Monolingual document retrieval for European languages. In *Information Retrieval*, 7(1-2):33–52, 2004. doi: 10.1023/B:INRT.0000009439.19151.4c. (Zitiert auf S. 162)

[Holmes 2001] Neville Holmes. Crouching Error, Hidden Markup. In *Computer*, 34(9), 2001. doi: 10.1109/2.947101. (Zitiert auf S. 35 und 105)

[Holt 1989] Patrik O'Brian Holt. Models of Writing: A Question of Interaction? In Noel Williams und Patrik O'Brian Holt (Hg.) *Computers and Writing: Models and tools*, Kap. 5, 50–60. Oxford, GB: Intellect, 1989. (Zitiert auf S. 23, 57 und 106)

[Holt 1992] Patrik O'Brian Holt. Preface. In Patrik O'Brian Holt und Noel Williams (Hg.) *Computers and Writing: State of the Art*, vii–x. Amsterdam, Niederlande: Kluwer, 1992. (Zitiert auf S. 117)

[Holt et al. 1990] Patrik O'Brian Holt, Dag C. Hegg und Terje Johnsen. Engineering Written Style. In *Computer Assisted Language Learning*, 2(1):27–35, 1990. (Zitiert auf S. 60)

[Horning 2006] Alice Horning. Professional Writers and Revision. In Alice Horning und Anne Becker (Hg.) *Revision: History, Theory, and Practice*, Reference Guides to Rhetoric and Composition, Kap. 8, 117–141. West Lafayette, IN, USA: Parlor Press, 2006. (Zitiert auf S. 78)

[Hoste et al. 2002] Véronique Hoste, Walter Daelemans, Iris Hendrickx und Antal van den Bosch. Evaluating the results of a memory-based word-expert approach to unrestricted word sense disambiguation. In *Proceedings of the ACL-02 workshop on Word sense disambiguation*, 95–101. Morristown, NJ, USA: Association for Computational Linguistics, 2002. doi: 10.3115/1118675.1118689. (Zitiert auf S. 164)

[Hull et al. 1987] Glynda Hull, Carolyn Ball, James L. Fox, Lori Levin und Deborah McCutchen. Computer detection of errors in natural language texts: Some research on pattern-matching. In *Computers and the Humanities*, 21(2):103–118, 1987. doi: 10.1007/BF00142750. (Zitiert auf S. 57, 97, 124 und 154)

[ISO 1998] ISO. ISO 9241-11:1998, Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability. Techn. Ber., International Organization for Standardization, 1998. (Zitiert auf S. 96 und 241)

[ISO 2001] ISO. ISO/IEC 9126-1:2001, Software engineering – Product quality – Part 1: Quality model. Techn. Ber., International Organization for Standardization, 2001. (Zitiert auf S. 97 und 241)

[Ivanova und Kübler 2008] Steliana Ivanova und Sandra Kübler. POS tagging for German: how important is the right context? In Nicoletta Calzolari, Khalid Choukri, Joseph M. Bente Maegaard, Jan Odjik, Stelios Piperidis und Daniel Tapias (Hg.) *LREC 2008 Sixth International Conference on Language Resources and Evaluation*. Paris, Frankreich: ELRA, 2008. (Zitiert auf S. 164)

[Janssen et al. 1996] Daniël Janssen, Luuk Van Waes und Huub Van den Bergh. Effects of Thinking Aloud on Writing Processes. In C. Michael Levy und Sarah Ransdell (Hg.) *The Science of Writing. Theories, Methods, Individual Differences, and Applications*, 233–250. Hillsdale, NJ, USA: Lawrence Erlbaum, 1996. (Zitiert auf S. 23)

[Johansson et al. 2008] Roger Johansson, Victoria Johansson, Åsa Wengelin und Kenneth Holmqvist. Reading during writing: Four different groups of writers. In *Lund Working Papers in Linguistics*, 53:43–59, 2008. (Zitiert auf S. 93, 109 und 116)

[Johansson et al. 2010] Roger Johansson, Åsa Wengelin, Victoria Johansson und Kenneth Holmqvist. Looking at the keyboard or the monitor: Relationship with text production processes. In *Reading and Writing*, 23(7):835–851, 2010. doi: 10.1007/s11145-009-9189-3. (Zitiert auf S. 79, 103, 104, 109, 115 und 116)

[Johnson und Guilfoyle 1989] Tim Johnson und Christine Guilfoyle. Commercial markets for NLP products. In Jeremy Peckham (Hg.) *Recent Developments and Applications of Natural Language Processing*, 1–7. London, GB: Kogan Page, 1989. (Zitiert auf S. 64, 70, 71, 121 und 122)

[Karttunen et al. 1998] Lauri Karttunen, Tamás Gaál und André Kempe. Xerox Finite-State Tool. Online, 1998. URL `http://www.cis.upenn.edu/~cis639/docs/xfst.html`. (Zitiert auf S. 132)

[Kellogg 1988] Ronald T. Kellogg. Attentional Overload and Writing Performance: Effects of Rough Draft and Outline Strategies. In *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14(2):355–356, 1988. (Zitiert auf S. 24)

[Kellogg 1996] Ronald T. Kellogg. A model of Working Memory in writing. In C. Michael Levy und Sarah Ransdell (Hg.) *The Science of Writing. Theories, Methods, Individual Differences and Applications*, 57–72. Hillsdale, NJ, USA: Lawrence Erlbaum, 1996. (Zitiert auf S. xv, 20 und 21)

[Kellogg 1999] Ronald T. Kellogg. Components of Working Memory in Text Production. In Mark Torrance und Gaynor Jeffery (Hg.) *The Cognitive Demands of Writing: Processing Capacity and Working Memory Effects in Text Production*, Bd. 3 von *Studies in Writing*, 42–61. Amsterdam, Niederlande: Amsterdam University Press, 1999. (Zitiert auf S. 24)

[Kellogg 2001] Ronald T. Kellogg. Commentary on part II: Processing modalities and development of expertise in writing. In Denis Alamargot und Lucile Chanquoy (Hg.) *Through the Models of Writing*, Bd. 9 von *Studies in Writing*, 219–228. Boston, Dordrecht, London: Kluwer, 2001. (Zitiert auf S. 24)

[Kellogg 2008] Ronald T. Kellogg. Training writing skills: A cognitive developmental perspective. In *Journal of Writing Research*, 1(1):1–26, 2008. (Zitiert auf S. 24 und 79)

[Kempen et al. 1986] Gerard Kempen, Gert Anbeek, Peter Desain, Leo Konst und Koenraad De Smedt. Author environments: Fifth generation text processors. In The Commission of the European Communities: Directorate General XIII: Telecommunications, Information, Industries & Innovation (Hg.) *ESPRIT'86 Results and Achievements*, 365–372. Amsterdam, New York, Oxford, Tokio: North-Holland, 1986. (Zitiert auf S. 63 und 83)

[Kempen und Vosse 1992] Gerard Kempen und Theo Vosse. A language-sensitive text editor for Dutch. In Patrik O'Brian Holt und Noel Williams (Hg.) *Computers and Writing: State of the Art*, 68–77. Boston, Dordrecht, London: Kluwer, 1992. (Zitiert auf S. 63 und 83)

[Kermes und Evert 2002] Hannah Kermes und Stefan Evert. YAC – a recursive chunker for unrestricted German text. In M. G. Rodriguez und C. P. Araujo (Hg.) *LREC 2002 Third International Conference on Language Resources and Evaluation*, 1805–1812. Paris, Frankreich: ELRA, 2002. (Zitiert auf S. 167, 170, 172 und 183)

[Khwaja und Urban 1993] Amir A. Khwaja und Joseph E. Urban. Syntax-directed editing environments: Issues and features. In *SAC '93: Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing*, 230–237. New York, NY, USA: ACM, 1993. doi: 10.1145/162754.162882. (Zitiert auf S. 80 und 188)

[Klenner et al. 2010a] Manfred Klenner, Angela Fahrni und Rico Sennrich. Real anaphora resolution is hard – the case of German. In Petr Sojka, Aleš Horák, Ivan Kopeček und Karel Pala (Hg.) *Text, Speech and Dialogue*, Bd. 6231 von *Lecture Notes in Computer Science*, Kap. 15, 109–116. Berlin, Heidelberg, New York: Springer, 2010. doi: 10.1007/978-3-642-15760-8_15. (Zitiert auf S. 161 und 221)

[Klenner et al. 2010b] Manfred Klenner, Don Tuggener, Angela Fahrni und Rico Sennrich. Anaphora Resolution with Real Preprocessing. In Hrafn Loftsson, Eiríkur Rögnvaldsson und Sigrún Helgadóttir (Hg.) *Advances in Natural Language Processing*, Bd. 6233 von *Lecture Notes in Computer Science*, Kap. 25, 215–225. Berlin, Heidelberg, New York: Springer, 2010. doi: 10.1007/978-3-642-14770-8_25. (Zitiert auf S. 161)

[Knobloch 2009] Clemens Knobloch. Noch einmal: Partikelverbkonstruktionen. In *Zeitschrift für Germanistische Linguistik*, 37(3):544–564, 2009. doi: 10.1515/ZGL.2009.035. (Zitiert auf S. 129)

[Ko et al. 2005] Andrew J. Ko, Htet H. Aung und Brad A. Myers. Design requirements for more flexible structured editors from a study of programmers' text editing. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, 1557–1560. New York, NY, USA: ACM, 2005. doi: 10.1145/1056808.1056965. (Zitiert auf S. 114)

[Kollberg 1998] Py Kollberg. *S-notation – a Computer Based Method for Studying and Representing Text Composition*. Masterarbeit, Kungliga Tekniska Högskolan Stockholm, Stockholm, Schweden, 1998. (Zitiert auf S. 41)

[Kollberg und Severinson Eklundh 2002] Py Kollberg und Kerstin Severinson Eklundh. Studying writers' revising patterns with S-notation analysis. In Thierry Olive und C. Michael Levy (Hg.) *Contemporary Tools and Techniques for Studying Writing*, Bd. 10 von *Studies in Writing*, 89–104. Boston, Dordrecht, London: Kluwer, 2002. (Zitiert auf S. 24, 41, 42 und 43)

[Koskenniemi 1983] Kimmo Koskenniemi. *Two-level Morphology: A General Computational Model for Word-Form Recognition and Production*. Dissertation, University of Helsinki, Helsinki, Finnland, 1983. (Zitiert auf S. 132)

[Koskenniemi und Haapalainen 1996] Kimmo Koskenniemi und Mariikka Haapalainen. GERTWOL – Lingsoft Oy. In Roland Hausser (Hg.) *Linguistische Verifikation. Dokumentation zur Ersten Morpholympics 1994*, Kap. 11, 121–140. Tübingen, Deutschland: Niemeyer, 1996. (Zitiert auf S. 126, 147, 152, 153 und 197)

[Kramer 2004] André Kramer. Rechtschreibkorrektursysteme im Vergleich. DITECT versus Microsoft Word. Online, 2004. URL http://www.mediensprache.net/de/networx/docs/networx-35.aspx. (Zitiert auf S. 66 und 68)

[Kruse et al. 2006] Otto Kruse, Katja Berger und Marianne Ulmi (Hg.) *Prozessorientierte Schreibdidaktik. Schreibtraining für Schule, Studium und Beruf*. Bern, Stuttgart, Wien: Haupt, 2006. (Zitiert auf S. 24)

[Kübler et al. 2010] Sandra Kübler, Kathrin Beck, Erhard W. Hinrichs und Heike Telljohann. Chunking German: An unsolved problem. In *Proceedings of the Fourth Linguistic Annotation Workshop*, 147–151. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (Zitiert auf S. 171 und 181)

[Kübler et al. 2006] Sandra Kübler, Erhard W. Hinrichs und Wolfgang Maier. Is it really that difficult to parse German? In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP 2006, Sydney, Australia, July 2006*. 2006. (Zitiert auf S. 165)

[Kurimo et al. 2009a] Mikko Kurimo, Sami Virpioja, Ville Turunen und Teemu Hirsimäki. Morpho Challenge – Evaluation of algorithms for unsupervised learning of morphology in various tasks and languages. In *Proceedings of the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL 2009*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009. (Zitiert auf S. 125)

[Kurimo et al. 2009b] Mikko Kurimo, Sami Virpioja, Ville T. Turunen, Graeme W. Blackwood und William Byrne. Overview and results of Morpho Challenge 2009. In *Working Notes for the CLEF 2009 Workshop*. 2009. (Zitiert auf S. 125)

[Lang 1986] Bernard Lang. On the usefulness of syntax directed editors. In *Advanced Programming Environments*, Bd. 244 von *Lecture Notes in Computer Science*, 47–51. Berlin, Heidelberg, New York: Springer, 1986. doi: 10.1007/3-540-17189-4_87. (Zitiert auf S. 80)

[Lanier 2010] Jaron Lanier. *You Are Not a Gadget: A Manifesto*. New York, NY, USA: Alfred A. Knopf, 2010. (Zitiert auf S. 189 und 190)

[Lapalme und Macklovitch 2008] Guy Lapalme und Elliott Macklovitch. Proposal for a Foreign Language Drafting Aid. In Robert Dale, Aurélien Max

und Michael Zock (Hg.) *LREC 2008 Workshop on NLP Resources, Algorithms and Tools for Authoring Aids*, 1–5. Paris, Frankreich: ELRA, 2008. (Zitiert auf S. 73)

[Largy et al. 2004] Pierre Largy, Lucile Chanquoy und A. Dédéyan. Orthographic revision: The case of subject-verb agreement in French. In Linda Allal, Lucile Chanquoy und Pierre Largy (Hg.) *Revision: Cognitive and Instructional Process*, Bd. 13 von *Studies in Writing*, 39–62. Boston, Dordrecht, London: Kluwer, 2004. (Zitiert auf S. 67)

[Leijten et al. 2010a] Mariëlle Leijten, Daniel Janssen und Luuk Van Waes. Error correction strategies of professional speech recognition users: Three profiles. In *Computers in Human Behavior*, 26(5):964–975, 2010. doi: 10.1016/j.chb.2010.02.010. (Zitiert auf S. 30, 72, 115 und 116)

[Leijten und Van Waes 2005a] Mariëlle Leijten und Luuk Van Waes. Inputlog: A logging tool for the research of writing processes. Techn. Ber., University of Antwerp, Faculty of Applied Economics, Department of Management, 2005. (Zitiert auf S. 43)

[Leijten und Van Waes 2005b] Mariëlle Leijten und Luuk Van Waes. Writing with speech recognition: The adaptation process of professional writers with and without dictating experience. In *Interacting with Computers*, 17(6):736–772, 2005. doi: 10.1016/j.intcom.2005.01.005. (Zitiert auf S. 115)

[Leijten et al. 2010b] Mariëlle Leijten, Luuk Van Waes und Sarah Ransdell. Correcting text production errors: Isolating the effects of writing mode from error span, input mode, and lexicality. In *Written Communication*, 27(2):189–227, 2010. doi: 10.1177/0741088309359139. (Zitiert auf S. 67 und 115)

[Lenders et al. 1996] Winfried Lenders, István Bátori, Grzegorz Dogil, Günther Görz und Uta Seewald. Stellungnahme der Jury für die Morpholympics 94. In Roland Hausser (Hg.) *Linguistische Verifikation. Dokumentation zur Ersten Morpholympics 1994*, 15–24. Tübingen, Deutschland: Niemeyer, 1996. (Zitiert auf S. 147)

[Levy und Ransdell 2002] C. Michael Levy und Sarah Ransdell. Writing with concurrent memory loads. In Thierry Olive und C. Michael Levy (Hg.) *Contemporary Tools and Techniques for Studying Writing*, Bd. 10 von *Studies in Writing*, 9–29. Boston, Dordrecht, London: Kluwer, 2002. (Zitiert auf S. 23)

[Lezius 1996] Wolfgang Lezius. Morphologiesystem MORPHY. In Roland Hausser (Hg.) *Linguistische Verifikation. Dokumentation zur Ersten Morpholympics 1994*, Kap. 5, 25–35. Tübingen, Deutschland: Niemeyer, 1996. (Zitiert auf S. 126)

[Lieberman et al. 2010] Michael D. Lieberman, Hanan Samet und Jagan Sankaranayananan. Geotagging: Using proximity, sibling, and prominence clues to understand comma groups. In *GIR '10: Proceedings of the 6th Workshop on Geographic Information Retrieval*, 1–8. New York, NY, USA: ACM, 2010. doi: 10.1145/1722080.1722088. (Zitiert auf S. 162)

[Lindgren 2005] Eva Lindgren. *Writing and revising: Didactic and Methodological Implications of Keystroke Logging*. Dissertation, Umeå Universitet, Umeå, Schweden, 2005. (Zitiert auf S. xv und 40)

[Lindgren und Sullivan 2006] Eva Lindgren und Kirk P. H. Sullivan. Analysing Online Revision. In Kirk P. H. Sullivan und Eva Lindgren (Hg.) *Computer Keystroke Logging and Writing: Methods and Applications*, Bd. 18 von *Studies in Writing*, Kap. 9, 157–188. Amsterdam, Niederlande: Elsevier Science, 2006. (Zitiert auf S. 40 und 92)

[Loftsson 2009] Hrafn Loftsson. Correcting a POS-tagged corpus using three complementary methods. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 523–531. Morristown, NJ, USA: Association for Computational Linguistics, 2009. (Zitiert auf S. 164)

[Lorenz 1996] Oliver Lorenz. *Automatische Wortformerkennung für das Deutsche im Rahmen von MALAGA*. Magisterarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, 1996. (Zitiert auf S. 126 und 130)

[Lüdeling 1999] Anke Lüdeling. *On Particle Verbs and Similar Constructions in German*. Stanford, CA, USA: Center for the Study of Language and Information, 1999. (Zitiert auf S. 129)

[Lutz 1983] Jean A. Lutz. *A Study of Professional and Experienced Writers Revising and Editing at the Computer and with Pen and Paper*. Dissertation, Rensselaer Polytechnic Institute, Troy, NY, USA, 1983. (Zitiert auf S. 23, 24, 28, 29, 33, 35, 38, 40 und 96)

[Maas 1996] Heinz D. Maas. MPRO – Ein System zur Analyse und Synthese deutscher Wörter. In Roland Hausser (Hg.) *Linguistische Verifikation. Dokumentation zur Ersten Morpholympics 1994*, Kap. 12, 141–166. Tübingen, Deutschland: Niemeyer, 1996. (Zitiert auf S. 127)

[Maas et al. 2009] Heinz D. Maas, Christoph Rösener und Axel Theofilidis. Morphosyntactic and semantic analysis of text: The MPRO tagging procedure. In Cerstin Mahlow und Michael Piotrowski (Hg.) *State of the Art in Computational Morphology: Workshop on Systems and Frameworks for Computational Morphology, SFCM 2009, Zurich, Switzerland, September 2009, Proceedings*, Bd. 41 von *Communications in Computer and Information Science*, Kap. 6, 76–87. Berlin, Heidelberg, New York: Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-04131-0_6. (Zitiert auf S. 127)

[Macdonald et al. 1982] Nina H. Macdonald, Lawrence T. Frase, Patricia S. Gingrich und Stacey A. Keenan. The Writer's Workbench: Computer aids for text analysis. In *IEEE Transactions on Communication*, 30(1):105–110, 1982. (Zitiert auf S. 48 und 58)

[Mahlow 2000] Cerstin Mahlow. *Automatische Wortformanalyse für das Spanische*. Magisterarbeit, Abteilung Computerlinguistik, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Deutschland, 2000. (Zitiert auf S. 94 und 125)

[Mahlow und Piotrowski 2009a] Cerstin Mahlow und Michael Piotrowski. SMM: Detailed, structured morphological analysis for Spanish. In *Polibits. Computer science and computer engineering with applications*, (39):41–48, 2009. (Zitiert auf S. 125, 127 und 155)

[Mahlow und Piotrowski 2009b] Cerstin Mahlow und Michael Piotrowski (Hg.). *State of the Art in Computational Morphology: Workshop on Systems and*

Frameworks for Computational Morphology, SFCM 2009, Zurich, Switzerland, September 2009, Proceedings, Bd. 41 von *Communications in Computer and Information Science*. Berlin, Heidelberg, New York: Springer, 2009. (Zitiert auf S. 184)

[Mahlow und Piotrowski 2009c] Cerstin Mahlow und Michael Piotrowski. A target-driven evaluation of morphological components for German. In Simon Clematide, Manfred Klenner und Martin Volk (Hg.) *Searching Answers – Festschrift in Honour of Michael Hess on the Occasion of his 60th Birthday*, 85–99. Münster, Deutschland: MV-Verlag, 2009. (Zitiert auf S. 148, 153 und 154)

[Marcu 2000] Daniel Marcu. The rhetorical parsing of unrestricted texts: A surface-based approach. In *Computational Linguistics*, 26(3):395–448, 2000. doi: 10.1162/089120100561755. (Zitiert auf S. 7)

[Max und Zock 2008] Aurélien Max und Michael Zock. Looking up phrase rephrasings via a pivot language. In *Coling 2008: Proceedings of the Workshop on Cognitive Aspects of the Lexicon (COGALEX 2008)*, 77–85. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008. (Zitiert auf S. 73)

[McClosky et al. 2006] David McClosky, Eugene Charniak und Mark Johnson. Effective Self-Training for Parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, 152–159. New York, NY, USA: Association for Computational Linguistics, 2006. (Zitiert auf S. 213)

[McCutchen 1996] Deborah McCutchen. A capacity theory of writing: Working memory in composition. In *Educational Psychology Review*, 8(3):299–325, 1996. doi: 10.1007/BF01464076. (Zitiert auf S. 24)

[McGee und Ericsson 2002] Tim McGee und Patricia Ericsson. The politics of the program: MS Word as the invisible grammarian. In *Computers and Composition*, 19(4):453–470, 2002. doi: 10.1016/S8755-4615(02)00142-1. (Zitiert auf S. 68 und 69)

[McGowan 1992] Steve McGowan. Ruskin to McRuskin – Degrees of interaction. In Patrik O'Brian Holt und Noel Williams (Hg.) *Computers and Writing: State of the Art*, 297–318. Boston, Dordrecht, London: Kluwer, 1992. (Zitiert auf S. 60)

[Merz-Grötsch 2005] Jasmin Merz-Grötsch. *Schreiben als System. Band 1: Schreibforschung und Schreibdidaktik. Ein Überblick*. Freiburg i. Breisgau, Deutschland: Filibach, 2. Aufl., 2005. (Zitiert auf S. 17 und 36)

[Meurers et al. 2010] Detmar Meurers, Ramon Ziai, Luiz Amaral, Adriane Boyd, Aleksandar Dimitrov, Vanessa Metcalf und Niels Ott. Enhancing Authentic Web Pages for Language Learners. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, 10–18. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (Zitiert auf S. 97, 124 und 154)

[Meyrowitz und van Dam 1982a] Norman Meyrowitz und Andries van Dam. Interactive editing systems: Part I. In *ACM Computing Surveys*, 14(3):321–352, 1982. doi: 10.1145/356887.356889. (Zitiert auf S. 48 und 80)

[Meyrowitz und van Dam 1982b] Norman Meyrowitz und Andries van Dam. Interactive editing systems: Part II. In *ACM Computing Surveys*, 14(3):353–415, 1982. doi: 10.1145/356887.356890. (Zitiert auf S. 6 und 87)

[Miller 1956] George A. Miller. The magical number seven plus or minus two: Some limits on our capacity for processing information. In *Psychological review*, 63(2):81–97, 1956. (Zitiert auf S. 30)

[Mißler 1997] Bettina Mißler. EUROJOB. Ein multilinguales Schreibwerkzeug. In Dagmar Knorr und Eva-Maria Jakobs (Hg.) *Textproduktion in elektronischen Umgebungen*, 157–170. Frankfurt/Main, Deutschland: Peter Lang, 1997. (Zitiert auf S. 37 und 38)

[Molitor-Lübbert 1997] Sylvie Molitor-Lübbert. Wissenschaftliche Textproduktion unter elektronischen Bedingungen. Ein heuristisches Modell der kognitiven Anforderungen. In Dagmar Knorr und Eva-Maria Jakobs (Hg.) *Textproduktion in elektronischen Umgebungen*, 47–66. Frankfurt/Main, Deutschland: Peter Lang, 1997. (Zitiert auf S. 18 und 19)

[Montero und Duque 2003] Juan M. Montero und M. Mar Duque. ANESTTE: A writer's assistant for a specific purpose language. In Dawn Archer, Paul Rayson, Andrew Wilson und Tony McEnery (Hg.) *Proceedings of the Corpus Linguistics 2003 conference*, 544–551. Lancaster University, 2003. (Zitiert auf S. 64, 71 und 72)

[Moore 1965] Gordon E. Moore. Cramming More Components onto Integrated Circuits. In *Electronics*, 38(8):114–117, 1965. doi: 10.1109/JPROC.1998.658762. (Zitiert auf S. 9)

[Moore 1975] Gordon E. Moore. Progress in digital integrated electronics. In *Electron Devices Meeting, 1975 International*, Bd. 21, 11–13. 1975. (Zitiert auf S. 9)

[Morris und Schwartz 1981] Joseph M. Morris und Mayer D. Schwartz. The design of a language-directed editor for block-structured languages. In *Proceedings of the ACM SIGPLAN SIGOA symposium on Text manipulation*, 28–33. New York, NY, USA: ACM, 1981. doi: 10.1145/800209.806451. (Zitiert auf S. 7 und 80)

[Mortenson 1987] Tom Mortenson. Writing style/readability checkers to add to your word processing. In *Computers and Composition*, 5(1):67–77, 1987. doi: 10.1016/S8755-4615(87)80017-8. (Zitiert auf S. 70)

[Müller 2004] Astrid Müller. Schreibenlernen in den Vorstellungen von Oberstufenschülern. Ergebnisse und Folgerungen aus einer Fallstudie. In Inge Blatt und Wilfried Hartmann (Hg.) *Schreibprozesse im medialen Wandel. Ein Studienbuch*, 141–159. Baltmannsweiler, Deutschland: Schneider Verlag Hohengehren, 2004. (Zitiert auf S. 102)

[Neal 1987] Lisa R. Neal. Cognition-sensitive design and user modeling for syntax-directed editors. In *CHI '87: Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface*, 99–102. New York, NY, USA: ACM, 1987. doi: 10.1145/29933.30866. (Zitiert auf S. 80)

[Negro und Chanquoy 1999] Isabelle Negro und Lucile Chanquoy. Subject – verb agreement errors in writing: Phonological and semantic control in adults. In Mark Torrance und Gaynor Jeffery (Hg.) *The Cognitive Demands of Writing:*

Processing Capacity and Working Memory Effects in Text Production, Bd. 3 von *Studies in Writing*, 83–98. Amsterdam, Niederlande: Amsterdam University Press, 1999. (Zitiert auf S. 1)

[Nicholson et al. 2008] Jeremy Nicholson, Valia Kordoni, Yi Zhang, Timothy Baldwin und Rebecca Dridan. Evaluating and extending the coverage of HPSG grammars: A case study for German. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis und Daniel Tapias (Hg.) *LREC 2008 Sixth International Conference on Language Resources and Evaluation*, 3134–3137. Paris, Frankreich: ELRA, 2008. (Zitiert auf S. 165)

[Niehues und Kolss 2009] Jan Niehues und Muntsin Kolss. A POS-based model for long-range reorderings in SMT. In *StatMT '09: Proceedings of the Fourth Workshop on Statistical Machine Translation*, 206–214. Morristown, NJ, USA: Association for Computational Linguistics, 2009. (Zitiert auf S. 162)

[Nielsen 1993] Jakob Nielsen. *Usability Engineering*. Burlington, MA, USA: Morgan Kaufmann, 1993. (Zitiert auf S. 96)

[Niessen und Ney 2000] Sonja Niessen und Hermann Ney. Improving SMT quality with morpho-syntactic analysis. In *Proceedings of the 18th conference on Computational linguistics*, 1081–1085. Morristown, NJ, USA: Association for Computational Linguistics, 2000. doi: 10.3115/992730.992809. (Zitiert auf S. 161)

[Norman 1981] Donald A. Norman. Categorization of action slips. In *Psychological Review*, 88:1–15, 1981. (Zitiert auf S. 5, 10, 11, 12, 13, 34, 78 und 238)

[Norman 1983] Donald A. Norman. Design rules based on analyses of human error. In *Communications of the ACM*, 26(4):254–258, 1983. doi: 10.1145/2163.358092. (Zitiert auf S. 10, 11, 12, 51 und 56)

[Norman 1988] Donald A. Norman. *The Psychology Of Everyday Things*. Basic Books, 1988. (Zitiert auf S. 22)

[Oakman 1994] Robert L. Oakman. The evolution of intelligent writing assistants: (Trends and future prospects. In *Sixth International Conference on Tools with Artificial Intelligence, 1994. Proceedings*, 233–234. 1994. doi: 10.1109/TAI.1994.346488. (Zitiert auf S. 70, 106 und 122)

[Ogren 2010] Philip Ogren. Improving Syntactic Coordination Resolution using Language Modeling. In *Proceedings of the NAACL HLT 2010 Student Research Workshop*, 1–6. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (Zitiert auf S. 213)

[Olive et al. 2009] Thierry Olive, Rui A. Alves und São L. Castro. Cognitive processes in writing during pause and execution periods. In *European Journal of Cognitive Psychology*, 21(5):758–785, 2009. doi: 10.1080/09541440802079850. (Zitiert auf S. 30)

[Olive et al. 2002] Thierry Olive, Ronald T. Kellogg und Annie Piolat. The triple task technique for studying the process of writing. In Thierry Olive und C. Michael Levy (Hg.) *Contemporary Tools and Techniques for Studying Writing*, Bd. 10 von *Studies in Writing*, 31–59. Boston, Dordrecht, London: Kluwer, 2002. (Zitiert auf S. 23 und 24)

[Perrin 2002] Daniel Perrin. Progression Analysis (PA): Investigating writing strategies in the workplace. In Thierry Olive und C. Michael Levy (Hg.) *Contemporary Tools and Techniques for Studying Writing*, Bd. 10 von *Studies in Writing*, 105–117. Boston, Dordrecht, London: Kluwer, 2002. (Zitiert auf S. 42 und 43)

[Perrin 2006a] Daniel Perrin. Progression analysis: An ethnographic computer-based multi-method approach for investigate natural writing processes. In Luuk Van Waes, Mariëlle Leijten und Christine M. Neuwirth (Hg.) *Writing and Digital Media*, Bd. 17 von *Studies in Writing*, 173–179. Amsterdam, Niederlande: Elsevier Science, 2006. (Zitiert auf S. 42 und 116)

[Perrin 2006b] Daniel Perrin. Schreibforschung im Kursalltag: Was die Progressionsanalyse praktisch nützt. In Otto Kruse, Katja Berger und Marianne Ulmi (Hg.) *Prozessorientierte Schreibdidaktik: Schreibtraining für Schule, Studium und Beruf*, 279–294. Bern, Stuttgart, Wien: Haupt, 2006. (Zitiert auf S. 3, 4, 42 und 116)

[Perrin und Wildi 2009] Daniel Perrin und Marc Wildi. Statistical Modeling of Writing Processes. In Charles Bazerman, Robert Krut, Karen Lunsford, Susan McLeod, Suzie Null, Paul Rogers und Amanda Stansell (Hg.) *Traditions of Writing Research*. New York, NY, USA: Routledge, 2009. (Zitiert auf S. 44)

[Pilotti et al. 2004] Maura Pilotti, Martin Chodorow und Kendell C. Thornton. Error detection in text: Do feedback and familiarity help? In *The Journal of general psychology*, 131(3):242–266, 2004. (Zitiert auf S. 67)

[Pilotti et al. 2006] Maura Pilotti, Kimberly Maxwell und Martin Chodorow. Does the effect of familiarity on proofreading change with encoding task and time? In *The Journal of general psychology*, 133(3):287–299, 2006. (Zitiert auf S. 67)

[Piolat 1991] Annie Piolat. Effects of word processing on text revision. In *Language and Education*, 5(4):255–272, 1991. (Zitiert auf S. 32, 33, 34, 37, 50 und 51)

[Piolat et al. 2004] Annie Piolat, Jean-Yves Roussey, Thierry Olive und Murielle Amada. Processing time and cognitive effort in revision: Effects of error type and of working memory capacity. In Linda Allal, Lucile Chanquoy und Pierre Largy (Hg.) *Revision. Cognitive and instructional processes*, Bd. 13 von *Studies in Writing*, 21–38. Boston, Dordrecht, London: Kluwer, 2004. (Zitiert auf S. 24, 37 und 109)

[Piolat et al. 1997] Annie Piolat, Jean-Yves Roussey und Olivier Thunin. Effects of screen presentation on text reading and revising. In *International Journal of Human-Computer Studies*, 47:565–589, 1997. (Zitiert auf S. 32, 34, 35 und 109)

[Puerta Melguizo et al. 2008a] Mari Carmen Puerta Melguizo, Teresa Bajo, Gracia Castillo, Olga Muñoz und Lou Boves. A proactive recommendation system for writing in the Internet age. In *Journal of Writing Research*, 2(1):65–81, 2008. (Zitiert auf S. 1 und 74)

[Puerta Melguizo et al. 2008b] Mari Carmen Puerta Melguizo, Olga Muñoz Ramos, Lou Boves, Toine Bogers und Antal van den Bosch. A personalized recommender system for writing in the Internet age. In Robert Dale, Aurélien Max und Michael Zock (Hg.) *LREC 2008 Workshop on NLP Resources,*

Algorithms and Tools for Authoring Aids, 21–26. Paris, Frankreich: ELRA, 2008. (Zitiert auf S. 74)

[Quinlan et al. 2009] Thomas Quinlan, Maaike Loncke, Mariëlle Leijten und Luuk Van Waes. Writers' shift between error correction and sentence composing: Competing and the executive function. Techn. Ber., University of Antwerp, 2009. (Zitiert auf S. 1, 24 und 115)

[Quirk et al. 1985] Randolph Quirk, Sidney Greenbaum, Geoffrey Leech und Jan Svartvik. *A Comprehensive Grammar of the English Language*. New York, NY, USA: Longman, 1985. (Zitiert auf S. 129)

[Rafferty und Manning 2008] Anna N. Rafferty und Christopher D. Manning. Parsing three German treebanks: Lexicalized and unlexicalized baselines. In *PaGe '08: Proceedings of the Workshop on Parsing German*, 40–46. Morristown, NJ, USA: Association for Computational Linguistics, 2008. (Zitiert auf S. 165)

[Ramshaw und Marcus 1995] Lance Ramshaw und Mitch Marcus. Text Chunking Using Transformation-Based Learning. In David Yarovsky und Kenneth Church (Hg.) *Proceedings of the Third Workshop on Very Large Corpora*, 82–94. Somerset, NJ, USA: Association for Computational Linguistics, 1995. (Zitiert auf S. 167 und 172)

[Ransdell und Levy 1996] Sarah Ransdell und C. Michael Levy. Working Memory Constraints on Writing Quality and Fluence. In C. Michael Levy und Sarah Ransdell (Hg.) *The Science of Writing. Theories, Methods, Individual Differences, and Applications*, 93–101. Hillsdale, NJ, USA: Lawrence Erlbaum, 1996. (Zitiert auf S. 24)

[Ransdell und Levy 1999] Sarah Ransdell und C. Michael Levy. Writing, Reading, and Speaking Memory Spans and the Importance of Resource Flexibility. In Mark Torrance und Gaynor Jeffery (Hg.) *The Cognitive Demands of Writing: Processing Capacity and Working Memory Effects in Text Production*, Bd. 3 von *Studies in Writing*, 99–113. Amsterdam, Niederlande: Amsterdam University Press, 1999. (Zitiert auf S. 1 und 24)

[Raskin 2000] Jef Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Reading, MA, USA: Addison-Wesley, 2000. (Zitiert auf S. 53, 86, 88, 91, 95, 97, 103, 105, 124 und 228)

[Rat für Deutsche Rechtschreibung 2006] Rat für Deutsche Rechtschreibung (Hg.) *Deutsche Rechtschreibung. Regeln und Wörterverzeichnis. Amtliche Regelung*. Tübingen, Deuschland: Gunter Narr, 2006. (Zitiert auf S. 129 und 146)

[Reiter und Dale 2000] Ehud Reiter und Robert Dale. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge, GB: Cambridge University Press, 2000. (Zitiert auf S. 18 und 22)

[Resnik 1999] Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. In *Journal of Artificial Intelligence Research*, 11:95–130, 1999. (Zitiert auf S. 213)

[Rhodes und Starner 1996] Bradley J. Rhodes und Thad Starner. Remembrance agent: A continuously running automated information retrieval system. In *Proceedings of the First International Conference on the Practical Application*

of Intelligent Agents and Multi Agent Technology (PAAM '96), 487–495. 1996. (Zitiert auf S. 73)

[Rijlaarsdam et al. 2004] Gert Rijlaarsdam, Michel Couzijn und Huub Van den Bergh. The study of revision as a writing process and as a learning-to-write process. Two prospective research agendas. In Linda Allal, Lucile Chanquoy und Pierre Largy (Hg.) *Revision. Cognitive and instructional processes*, Bd. 13 von *Studies in Writing*, 189–2007. Boston, Dordrecht, London: Kluwer, 2004. (Zitiert auf S. 96)

[van Rijsbergen 1979] Cornelis J. van Rijsbergen. *Information Retrieval*. London, GB: Butterworths, 1979. (Zitiert auf S. 97)

[Rimrott und Heift 2008] Anne Rimrott und Trude Heift. Evaluating automatic detection of misspellings in German. In *Language Learning & Technology*, 12(3):73–92, 2008. (Zitiert auf S. 67)

[Rose 1983] Mike Rose. *Writer's Block: The Cognitive Dimension*. Studies in Writing and Rhetoric. Carbondale, IL, USA: Southern Illinois University Press, 1983. (Zitiert auf S. 69)

[Ross 1991] Donald Ross. Prospects for Writer's Workstations in the Coming Decade. In Gail E. Hawisher und Cynthia L. Selfe (Hg.) *Evolving Perspectives on Computers and Composition Studies*, 84–110. Urbana, IL, USA: National Council of Teachers of English, 1991. (Zitiert auf S. 65)

[Rosson 1983] Mary B. Rosson. Patterns of experience in text editing. In *CHI '83: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 171–175. New York, NY, USA: ACM, 1983. doi: 10.1145/800045.801604. (Zitiert auf S. 44, 51 und 114)

[Rozovskaya und Roth 2010] Alla Rozovskaya und Dan Roth. Training Paradigms for Correcting Errors in Grammar and Usage. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 154–162. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (Zitiert auf S. 71)

[Sanford und Moxey 1989] Anthony J. Sanford und Linda M. Moxey. Language Understanding and the Cognitive Ergonomics of Style. In Noel Williams und Patrik O'Brian Holt (Hg.) *Computers and Writing: Models and tools*, Kap. 4, 38–49. Oxford, GB: Intellect, 1989. (Zitiert auf S. 67)

[Scardamalia und Bereiter 1983] Marlene Scardamalia und Carl Bereiter. The development of evaluative, diagnostic and remedial capabilities in children's composing. In Margaret Martlew (Hg.) *The psychology of written language. Developmental and educational perspectives*, 67–95. New York, NY, USA: Wiley and Sons, 1983. (Zitiert auf S. xv und 29)

[Schiehlen 2002] Michael Schiehlen. Experiments in German noun chunking. In *Proceedings of the 19th international conference on Computational Linguistics*, 1–7. Morristown, NJ, USA: Association for Computational Linguistics, 2002. doi: 10.3115/1072228.1072349. (Zitiert auf S. 167 und 183)

[Schiller et al. 1999] Anne Schiller, Simone Teufel, Christine Stöckert und Christine Thielen. Guidelines für das Tagging deutscher Textcorpora mit STTS. Techn. Ber., Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, Stuttgart, Deutschland, 1999. (Zitiert auf S. 164)

[Schmid 1994] Helmut Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, 44–49. 1994. (Zitiert auf S. 139 und 162)

[Schmid 1995] Helmut Schmid. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*, 47–50. 1995. (Zitiert auf S. 162 und 167)

[Schmid 2008] Helmut Schmid. Tokenizing and part-of-speech tagging. In Anke Lüdeling und Merja Kytö (Hg.) *Corpus Linguistics. An International Handbook*, Bd. 1 von *Handbooks of Linguistics and Communication Science*, Kap. 24, 527–551. Berlin, Deutschland: Mouton de Gruyter, 2008. doi: 10.1515/9783110211429.4.527. (Zitiert auf S. 162 und 165)

[Schmid et al. 2004] Helmut Schmid, Arne Fitschen und Ulrich Heid. A German computational morphology covering derivation, composition, and inflection. In *Proceedings of the IVth International Conference on Language Resources and Evaluation (LREC 2004)*, 1263–1266. 2004. (Zitiert auf S. 131)

[Schmid und Schulte im Walde 2000] Helmut Schmid und Sabine Schulte im Walde. Robust German noun chunking with a probabilistic context-free grammar. In *Proceedings of the 18th conference on Computational linguistics*, 726–732. Morristown, NJ, USA: Association for Computational Linguistics, 2000. doi: 10.3115/992730.992751. (Zitiert auf S. 167 und 172)

[Schmitz und Zöllner 2006] Martina Schmitz und Nicole Zöllner. Der rote Faden zur Fach- oder Maturaarbeit – Eine Simulation. In Otto Kruse, Katja Berger und Marianne Ulmi (Hg.) *Prozessorientierte Schreibdidaktik: Schreibtraining für Schule, Studium und Beruf*, 69–91. Bern, Stuttgart, Wien: Haupt, 2006. (Zitiert auf S. 37)

[Schulze 2004] Markus Schulze. *Ein sprachunabhängiger Ansatz zur Entwicklung deklarativer, robuster LA-Grammatiken mit einer exemplarischen Anwendung auf das Deutsche und das Englische*. Dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Deutschland, 2004. (Zitiert auf S. 126)

[Schumacher et al. 2004] Helmut Schumacher, Jacqueline Kubczak, Renate Schmidt und Vera de Ruiter. *VALBU – Valenzwörterbuch deutscher Verben*. Studien zur deutschen Sprache. Tübingen, Deutschland: Gunter Narr, 2004. (Zitiert auf S. 128)

[Schwitter et al. 2000] Rolf Schwitter, Diego Mollá, Rachel Fournier und Michael Hess. Answer extraction towards better evaluations of NLP systems. In *ANLP/NAACL 2000 Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems - Volume 6*, 20–27. Morristown, NJ, USA: Association for Computational Linguistics, 2000. doi: 10.3115/1117595.1117599. (Zitiert auf S. 9)

[Sennrich et al. 2009] Rico Sennrich, Gerold Schneider, Martin Volk und Martin Warin. A new hybrid dependency parser for German. In Christian Chiarcos, Richard Eckart de Castilho und Manfred Stede (Hg.) *Von der Form zur Bedeutung: Texte automatisch verarbeiten / From Form to Meaning: Processing Texts Automatically. Proceedings of the Biennial GSCL Conference 2009*, 115–124. Tübingen, Deutschland: Gunter Narr, 2009. (Zitiert auf S. 161)

[Severinson Eklundh 1992] Kerstin Severinson Eklundh. Problems in Achieving a Global Perspective of the Text in Computer-based Writing. In Mike Sharples (Hg.) *Computers and Writing: Issues and Implementation*, 73–84. Boston, Dordrecht, London: Kluwer, 1992. (Zitiert auf S. 35)

[Severinson Eklundh 1994] Kerstin Severinson Eklundh. Linear and nonlinear strategies in computer-based writing. In *Computers and Composition*, 11(3):203–216, 1994. doi: 10.1016/8755-4615(94)90013-2. (Zitiert auf S. 24, 32, 36, 41, 42, 96 und 109)

[Severinson Eklundh und Kollberg 1996] Kerstin Severinson Eklundh und Py Kollberg. A Computer Tool and Framework for Analyzing Online Revisions. In C. Michael Levy und Sarah Ransdell (Hg.) *The Science of Writing. Theories, Methods, Individual Differences, and Applications*, 163–188. Hillsdale, NJ, USA: Lawrence Erlbaum, 1996. (Zitiert auf S. 14, 41, 84 und 117)

[Sharples 1994] Mike Sharples. Computer support for the rhythms of writing. In *Computers and Composition*, 11(3):217–226, 1994. doi: 10.1016/8755-4615(94)90014-0. (Zitiert auf S. 24 und 105)

[Sharples 1996a] Mike Sharples. An Account of Writing as Creative Design. In C. Michael Levy und Sarah Ransdell (Hg.) *The Science of Writing. Theories, Methods, Individual Differences, and Applications*, 127–148. Hillsdale, NJ, USA: Lawrence Erlbaum, 1996. (Zitiert auf S. 2 und 110)

[Sharples 1996b] Mike Sharples. Design for New Writing Environments. In Mike Sharples und Thea van der Geest (Hg.) *The New Writing Environment: Writers at Work in a World of Technology*, Kap. 7, 97–115. Berlin, Heidelberg, New York: Springer, 1996. (Zitiert auf S. 98, 106 und 107)

[Sharples 1999] Mike Sharples. *How We Write: Writing As Creative Design*. New York, NY, USA: Routledge, 1999. (Zitiert auf S. xv, 2, 20, 22, 30, 57, 78 und 93)

[Sharples und van der Geest 1996] Mike Sharples und Thea van der Geest. *The New Writing Environment: Writers at Work in a World of Technology*. Berlin, Heidelberg, New York: Springer, 1996. (Zitiert auf S. 106)

[Sharples et al. 1989] Mike Sharples, James Goodlet und Lyn Pemberton. Developing a Writer's Assistant. In Noel Williams und Patrik O'Brian Holt (Hg.) *Computers and Writing: Models and tools*, Kap. 3, 22–37. Oxford, GB: Intellect, 1989. (Zitiert auf S. 60, 61 und 206)

[Sharples und Pemberton 1990] Mike Sharples und Lyn Pemberton. Starting from the Writer: Guidelines for the Design of User-Centred Document Processors. In *Computer Assisted Language Learning*, 2(1):37–57, 1990. (Zitiert auf S. 50, 60, 61 und 106)

[Smithson 1986] Isaiah Smithson. The Writing Workshop. In *Computers and Composition*, 4(1):78–94, 1986. doi: 10.1016/S8755-4615(86)80009-3. (Zitiert auf S. 43, 48, 58 und 64)

[Sommers 1980] Nancy Sommers. Revision Strategies of Student Writers and Experienced Adult Writers. In *College Composition and Communication*, 31(4):378–388, 1980. doi: 10.2307/356588. (Zitiert auf S. 28, 38, 39, 89 und 92)

[Stallman 1981] Richard M. Stallman. EMACS. The extensible, customizable self-documenting display editor. In *Proceedings of the ACM SIGPLAN SIGOA symposium on Text manipulation*, 147–156. New York, NY, USA: ACM, 1981. doi: 10.1145/800209.806466. (Zitiert auf S. 188 und 195)

[Stein et al. 2010] Benno Stein, Martin Potthast und Martin Trenkmann. Retrieving Customary Web Language to Assist Writers. In Cathal Gurrin, Yulan He, Gabriella Kazai, Udo Kruschwitz, Suzanne Little, Thomas Roelleke, Stefan Rüger und Keith van Rijsbergen (Hg.) *Advances in Information Retrieval, Proceedings of the 32nd European Conference on Information Retrieval (ECIR 2010)*, Bd. 5993 von *Lecture Notes in Computer Science*, 631–635. Berlin, Heidelberg, New York: Springer, 2010. doi: http://dx.doi.org/10.1007/978-3-642-12275-0_64. (Zitiert auf S. 73)

[Stephenson 1999] Neal Stephenson. *In the Beginning…was the Command Line*. New York, NY, USA: Harper, 1999. (Zitiert auf S. 110, 111 und 189)

[Strömfors und Jonesjö 1981] Ola Strömfors und Lennat Jonesjö. The implementation and experiences of a structure-oriented text editor. In *SIGPLAN Notices*, 16(6):22–27, 1981. doi: 10.1145/872730.806450. (Zitiert auf S. 81)

[Stutz 2007] Michael Stutz. Explore powerful UNIX writer's tools. Using new, open source equivalents of the classic UNIX Writer's Workbench. Online, 2007. URL `http://www.ibm.com/developerworks/edu/au-dw-au-writersworkbench-i.html`. (Zitiert auf S. 58)

[Sullivan und Lindgren 2006] Kirk P. H. Sullivan und Eva Lindgren (Hg.) *Computer Key-Stroke Logging and Writing: Methods and Applications*, Bd. 18 von *Studies in Writing*. Amsterdam, Niederlande: Elsevier Science, 2006. (Zitiert auf S. 43 und 109)

[Sweller 1999] John Sweller. *Instructional Design in Technical Areas*, Bd. 43 von *Australian Educational Review*. Camberwell, Vic., Australien: ACER Press, 1999. (Zitiert auf S. 30)

[Taylor 1987] Lee R. Taylor. Software Views: A Fistful of Word-Processing Programs. In *Computers and Composition*, 5(1):79–90, 1987. (Zitiert auf S. 18, 25, 55 und 104)

[Teitelbaum und Reps 1981] Tim Teitelbaum und Thomas Reps. The Cornell Program Synthesizer: A syntax-directed programming environment. In *Communications of the ACM*, 24(9):563–573, 1981. doi: 10.1145/358746.358755. (Zitiert auf S. 79 und 80)

[Telljohann et al. 2009] Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, Heike Zinsmeister und Kathrin Beck. Stylebook for the Tübingen treebank of written German (TüBa-D/Z). Techn. Ber., Universität Tübingen, Seminar für Sprachwissenschaft, 2009. (Zitiert auf S. 164, 165 und 200)

[Tjong Kim Sang und Buchholz 2000] Erik F. Tjong Kim Sang und Sabine Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*, 127–132. Morristown, NJ, USA: Association for Computational Linguistics, 2000. doi: 10.3115/1117601.1117631. (Zitiert auf S. 167 und 169)

[Torrance und Galbraith 2006] Mark Torrance und David Galbraith. The processing demands of writing. In Charles A. MacArthur, Steve Graham und Jill Fitzgerald (Hg.) *Handbook of Writing Research*, 67–82. New York, NY, USA: Guilford, 2006. (Zitiert auf S. 24 und 37)

[Torrance und Jeffery 1999] Mark Torrance und Gaynor Jeffery. Writing processes and cognitive demands. In Mark Torrance und Gaynor Jeffery (Hg.) *The Cognitive Demands of Writing: Processing Capacity and Working Memory Effects in Text Production*, Bd. 3 von *Studies in Writing*. Amsterdam, Niederlande: Amsterdam University Press, 1999. (Zitiert auf S. 25 und 79)

[Van De Vanter 1992] Michael L. Van De Vanter. *User Interaction in Language-Based Editing Systems*. Dissertation, University of California, Berkeley, Berkeley, CA, USA, 1992. (Zitiert auf S. 95)

[Van De Vanter 1995] Michael L. Van De Vanter. Practical language-based editing for software engineers. In *Software Engineering and Human-Computer Interaction*, Bd. 896 von *Lecture Notes in Computer Science*, 251–267. Berlin, Heidelberg, New York: Springer, 1995. doi: 10.1007/BFb0035821. (Zitiert auf S. 7, 80, 93, 94 und 171)

[Van De Vanter und Boshernitsan 2000] Michael L. Van De Vanter und Marat Boshernitsan. Displaying and Editing Source Code in Software Engineering Environments. In *Second International Symposium on Constructing Software Engineering Tools (CoSET'2000)*. 2000. (Zitiert auf S. xvi, 7, 80, 84, 86, 93 und 95)

[Van De Vanter et al. 1992] Michael L. Van De Vanter, Susan L. Graham und Robert A. Ballance. Coherent user interfaces for language-based editing systems. In *International Journal of Man-Machine Studies*, 37(4):431–466, 1992. doi: 10.1016/0020-7373(92)90004-5. (Zitiert auf S. 80, 81 und 84)

[Van Ittersum 2008] Derek Van Ittersum. Computing Attachments: Engelbart's Controversial Writing Technology. In *Computers and Composition*, 25(2):143–164, 2008. doi: 10.1016/j.compcom.2007.12.001. (Zitiert auf S. 49 und 105)

[Van Waes 1992] Luuk Van Waes. The influence of the computer on writing profiles. In Henk P. Maat und Michaël Steehouder (Hg.) *Studies of functional text quality*, 173–186. Amsterdam, Niederlande: Rodopi, 1992. (Zitiert auf S. 110 und 117)

[Van Waes et al. 2006] Luuk Van Waes, Mariëlle Leijten und Christine M. Neuwirth (Hg.) *Writing and Digital Media*, Bd. 17 von *Studies in Writing*. Amsterdam, Niederlande: Elsevier Science, 2006. (Zitiert auf S. 43 und 109)

[Van Waes et al. 2010] Luuk Van Waes, Mariëlle Leijten und Thomas Quinlan. Reading during sentence composing and error correction: A multilevel analysis of the influences of task complexity. In *Reading and Writing*, 23(7):803–834, 2010. doi: 10.1007/s11145-009-9190-x. (Zitiert auf S. 24, 30, 93 und 116)

[Van Waes et al. 2009] Luuk Van Waes, Mariëlle Leijten und Daphne Van Weijen. Keystroke logging in writing research: Observing writing processes with Inputlog. In *GFL – German as a foreign language*, (2–3):41–64, 2009. (Zitiert auf S. 43, 109 und 116)

[Van Waes und Schellens 2003] Luuk Van Waes und Peter J. Schellens. Writing profiles: The effect of the writing mode on pausing and revision patterns of experienced writers. In *Journal of Pragmatics*, 35(6):829–853, 2003. doi: 10.1016/S0378-2166(02)00121-2. (Zitiert auf S. 93 und 117)

[Vernon 2000] Alex Vernon. Computerized grammar checkers 2000: Capabilities, limitations, and pedagogical possibilities. In *Computers and Composition*, 17(3):329–349, 2000. doi: 10.1016/S8755-4615(00)00038-4. (Zitiert auf S. 49, 64, 65, 66, 68, 69 und 70)

[Waloszek 2008a] Gerd Waloszek. Waiting at the Computer: Busy Indicators and System Feedback – Part 1. Online, 2008. URL `http://www.sapdesignguild.org/community/design/busy_feedback_1.asp`. (Zitiert auf S. 124)

[Waloszek 2008b] Gerd Waloszek. Waiting at the Computer: Busy Indicators and System Feedback – Part 3. Online, 2008. URL `http://www.sapdesignguild.org/community/design/busy_feedback_3.asp`. (Zitiert auf S. 124)

[Weder 2010] Mirjam Weder. Keystroke-Logging und Stimulated-Recall in der Orthographie-Forschung. In *Bulletin suisse de linguistique appliquée*, (91):85–104, 2010. (Zitiert auf S. 42 und 115)

[Weimer et al. 2007] Markus Weimer, Iryna Gurevych und Max Mühlhäuser. Automatically assessing the post quality in online discussions on software. In *ACL '07: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, 125–128. Morristown, NJ, USA: Association for Computational Linguistics, 2007. (Zitiert auf S. 162)

[Wengelin et al. 2010] Åsa Wengelin, Mariëlle Leijten und Luuk Van Waes. Studying reading during writing: New perspectives in research. In *Reading and Writing*, 23(7):735–742, 2010. doi: 10.1007/s11145-009-9187-5. (Zitiert auf S. 1, 21, 115 und 116)

[Wengelin et al. 2009] Åsa Wengelin, Mark Torrance, Kenneth Holmqvist, Sol Simpson, David Galbraith, Victoria Johansson und Roger Johansson. Combined eyetracking and keystroke-logging methods for studying cognitive processes in text production. In *Behavior Research Methods*, 41(2):337–351, 2009. doi: 10.3758/BRM.41.2.337. (Zitiert auf S. 30, 93 und 116)

[Whiteside et al. 1982] John Whiteside, Norman Archer, Dennis Wixon und Michael Good. How do people really use text editors? In *ACM SIGOA Newsletter*, 3(1-2):29–40, 1982. doi: 10.1145/966873.806474. (Zitiert auf S. 44, 50, 51, 81, 87 und 114)

[Whithaus 2004] Carl Whithaus. The Development of Early Computer-Assisted Writing Instruction (1960–1978): The Double Logic of Media and Tools. In *Computers and the Humanities*, 38(2):149–162, 2004. doi: 10.1023/B:CHUM.0000031171.79841.02. (Zitiert auf S. 70)

[Wilcox-O'Hearn et al. 2008] L. Amber Wilcox-O'Hearn, Graeme Hirst und Alexander Budanitsky. Real-word spelling correction with trigrams: A reconsideration of the Mays, Damerau, and Mercer model. In Alexander Gelbukh (Hg.) *Proceedings, 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2008) (Lecture Notes in*

Computer Science 4919, Springer-Verlag), 605–616. Berlin, Heidelberg, New York, 2008. (Zitiert auf S. 66)

[Williams 1989] Noel Williams. Computer assisted writing software: RUSKIN. In Noel Williams und Patrik O'Brian Holt (Hg.) *Computers and Writing: Models and tools*, Kap. 1, 1–16. Oxford, GB: Intellect, 1989. (Zitiert auf S. 59, 60, 106 und 107)

[Williams 1990a] Noel Williams. Editorial. In *Computer Assisted Language Learning*, 2(1):1–4, 1990. (Zitiert auf S. 59)

[Williams 1990b] Noel Williams. Writers' Problems and Computer Solutions. In *Computer Assisted Language Learning*, 2(1):5–25, 1990. (Zitiert auf S. 59, 60, 70, 98, 106 und 107)

[Williams 1992] Noel Williams. New technologies. New writing. New problems? In Patrik O'Brian Holt und Noel Williams (Hg.) *Computers and Writing: State of the Art*, Kap. 1, 1–19. Boston, Dordrecht, London: Kluwer, 1992. (Zitiert auf S. 114)

[Williams 1993] Noel Williams. Postwriting software and the classroom. In Moira Monteith (Hg.) *Computers and Language*, 114–123. Oxford, GB: Intellect, 1993. (Zitiert auf S. 59)

[Williams und Holt 1989] Noel Williams und Patrik O'Brian Holt (Hg.) *Computers And Writing: Models And Tools*. Oxford, GB: Intellect, 1989. (Zitiert auf S. 18 und 64)

[Willis 2008] Timothy Willis. *A flexible expansion algorithm for user-chosen abbreviations*. Dissertation, University of Edinburgh, Edinburgh, GB, 2008. (Zitiert auf S. 73)

[Willis et al. 2002] Timothy Willis, Helen Pain, Shari Trewin und Stephen Clark. Informing Flexible Abbreviation Expansion for Users with Motor Disabilities. In Klaus Miesenberger, Joachim Klaus und Wolfgang Zagler (Hg.) *Computers Helping People with Special Needs, 8th International Conference, ICCHP 2002, Linz, Austria, July 2002, Proceedings*, Bd. 2398 von *Lecture Notes in Computer Science*, 251–258. Berlin, Heidelberg, New York: Springer, 2002. (Zitiert auf S. 72)

[Wirth 1971] Niklaus Wirth. Program development by stepwise refinement. In *Communications of the ACM*, 14(4):221–227, 1971. doi: 10.1145/362575.362577. (Zitiert auf S. 80)

[Wohl 2006] Amy D. Wohl. How we process words: The marketing of WP software. In *IEEE Annals of the History of Computing*, 28(4):88–91, 2006. doi: 10.1109/MAHC.2006.66. (Zitiert auf S. 48 und 49)

[Wood 1981] Steven R. Wood. Z – the 95% program editor. In *SIGPLAN Notices*, 16(6):1–7, 1981. doi: 10.1145/872730.806447. (Zitiert auf S. 5, 50, 80, 81, 190 und 222)

[Yang 1989] Yiya Yang. *User-oriented design of Undo support*. Dissertation, Heriot-Watt University, Edinburgh, GB, 1989. (Zitiert auf S. 87)

[Yang 1990] Yiya Yang. Experimental rapid prototype of undo support. In *Information and Software Technology*, 32(9):625–635, 1990. doi: 10.1016/0950-5849(90)90208-9. (Zitiert auf S. 87)

[Yang 1992] Yiya Yang. Supporting Writing with an Undo Mechanism. In Patrik O'Brian Holt und Noel Williams (Hg.) *Computers and Writing: State of the Art*, 187–196. Boston, Dordrecht, London: Kluwer, 1992. (Zitiert auf S. 87)

[Zavrel und Daelemans 1999] Jakub Zavrel und Walter Daelemans. Recent Advances in Memory-Based Part-of-Speech Tagging. In *Actas del VI Simposio Internacional de Comunicación Social*, 590–597. 1999. (Zitiert auf S. 163 und 200)

[Zielinski und Simon 2008] Andrea Zielinski und Christian Simon. Morphisto: An Open-Source Morphological Analyzer for German. In *Proceedings of the FSMNLP 2008: Seventh International Workshop on Finite-State Methods and Natural Language Processing*, 177–184. 2008. (Zitiert auf S. 126)

[Zielinski et al. 2009] Andrea Zielinski, Christian Simon und Tilman Wittl. Morphisto: Service-oriented open source morphology for German. In Cerstin Mahlow und Michael Piotrowski (Hg.) *State of the Art in Computational Morphology*, Bd. 41 von *Communications in Computer and Information Science*, Kap. 5, 64–75. Berlin, Heidelberg, New York: Springer, 2009. doi: 10.1007/978-3-642-04131-0_5. (Zitiert auf S. 126)

[Zifonun et al. 1997] Gisela Zifonun, Ludger Hoffmann und Bruno Strecker. *Grammatik der Deutschen Sprache*. Berlin, Deutschland: Walter de Gruyter, 1997. (Zitiert auf S. 128)

[Zock 1985] Michael Zock. Le fil d'ariane ou les grammaires de texte comme guide dans l'organisation et l'expression de la pensée en langue maternelle et/ou étrangère. Techn. Ber., UNESCO, 1985. (Zitiert auf S. 5, 17 und 18)

[Zock 1998] Michael Zock. La rédaction: Conflit entre les idées et leur organisation. In *Proceedings of the fifth conference TALN 1998 (Traitement Automatique des Langues Naturelles)*, 245–248. 1998. (Zitiert auf S. 29 und 30)

[Zock 1999] Michael Zock. Holmes meets Montgomery: An unusual yet necessary encounter between a detective and a general, or, the need of analytical and strategic skills in outline planning. In *VI Simposio Internacional de Comunicacion Social*, 478–483. 1999. (Zitiert auf S. 18)

[Zock 2001] Michael Zock. Learn to speak and to write, learn to use your mind. Invited Talk at 8th European Workshop on Natural Language Generation, ACL 2001, 2001. (Zitiert auf S. 18)

[Zock et al. 2004] Michael Zock, Nicolas Hernandez und Aurélien Max. L'ordinateur à la rescousse du rédacteur: Comment reconnaître automatiquement des liens entre des idées afin de construire un plan? In *La génération de langue naturelle. Journée d'étude ATALA*, 41–44. 2004. (Zitiert auf S. 18)

Eigene Publikationen und Vorträge im Rahmen dieser Arbeit

Cerstin Mahlow (2008a). Dieser Satz kein Verb. In *readme. Das Bulletin der Alumni Wirtschaftsinformatik Universität Zürich*, 20:11.

Cerstin Mahlow (2008b). Wort oder Word? Beeinflusst die Abstraktionsebene den Editierprozess? Eingeladener Vortrag an der Abteilung Computerlinguistik, Friedrich-Alexander-Universität Erlangen-Nürnberg.

Cerstin Mahlow (2010). Writing research and natural language processing: Challenges and opportunities. Vortrag auf der SIG Writing Conference 2010.

Cerstin Mahlow und Michael Piotrowski (2008a). Computational linguistics for word processing: Opportunities and limits. In Sylvana Sofkova Hashemi, Ola Knutsson, Rickard Domeij und Sofie Johansson Kokkinakis (Hg.) *Workshop on NLP for Reading and Writing: Resources, Algorithms and Tools*. KTH Royal Institute of Technology.

Cerstin Mahlow und Michael Piotrowski (2008b). Linguistic support for revising and editing. In Alexander Gelbukh (Hg.) *Computational Linguistics and Intelligent Text Processing: 9th International Conference, CICLing 2008, Haifa, Israel, February 17–23, 2008. Proceedings*, 631–642. Berlin, Heidelberg, New York: Springer.

Cerstin Mahlow und Michael Piotrowski (2009a). LingURed: Language-Aware Editing Functions Based on NLP Resources. In *Proceedings of the International Multiconference on Computer Science and Information Technology*, Bd. 4, 243–250. Polish Information Processing Society.

Cerstin Mahlow und Michael Piotrowski (2009b). Making word processors process words. Talk at the onsite sessions (June 18 – 21) and online discussion forum in Sakai (February 23 – 27) at Computers and Writing 2009.

Cerstin Mahlow und Michael Piotrowski (2009c). Opportunities and limits for language awareness in text editors. In Rickard Domeij, Sofie J. Kokkinakis, Ola Knutsson und Sylvana Sofkova Hashemi (Hg.) *Proceedings of the Workshop on NLP for Reading and Writing – Resources, Algorithms and Tools (SLTC 2008)*, Bd. 3 von *NEALT Proceedings Series*, 14–18. Tartu University Library (Estonia). URL http://hdl.handle.net/10062/8696.

Cerstin Mahlow und Michael Piotrowski (Hg.) (2009d). *State of the Art in Computational Morphology: Workshop on Systems and Frameworks for Computational Morphology, SFCM 2009, Zurich, Switzerland, September 2009, Proceedings*, Bd. 41 von *Communications in Computer and Information Science*. Berlin, Heidelberg, New York: Springer.

Cerstin Mahlow und Michael Piotrowski (2009e). A target-driven evaluation of morphological components for German. In Simon Clematide, Manfred Klenner und Martin Volk (Hg.) *Searching Answers – Festschrift in Honour of Michael Hess on the Occasion of his 60th Birthday*, 85–99. Münster, Deutschland: MV-Verlag.

Cerstin Mahlow und Michael Piotrowski (2010). Noun phrase chunking and categorization for authoring aids. In Manfred Pinkal, Ines Rehbein, Sabine Schulte im Walde und Angelika Storrer (Hg.) *Semantic Approaches in Natural Language Processing: Proceedings of the Conference on Natural Language Processing 2010 (KONVENS)*, 57–65. Saarbrücken, Deutschland: Universaar.

Cerstin Mahlow und Michael Piotrowski (2011a). Lingured: Linguistisch unterstütztes Redigieren. Poster auf der DGfS Jahrestagung 2011.

Cerstin Mahlow und Michael Piotrowski (2011b). Writing and writing tools: Separate worlds? Vortrag auf 4th International Conference on Writing Research (Writing Research Across Borders II).

Cerstin Mahlow, Michael Piotrowski und Michael Hess (2008). Language-aware text editing. In Robert Dale, Aurélien Max und Michael Zock (Hg.) *LREC 2008 Workshop on NLP Resources, Algorithms and Tools for Authoring Aids*, 9–13. Paris, Frankreich: ELRA.

Michael Piotrowski und Cerstin Mahlow (2009). Linguistic editing support. In *DocEng'09: Proceedings of the 2009 ACM Symposium on Document Engineering*, 214–217. New York, NY, USA: ACM.

Michael Piotrowski und Cerstin Mahlow (2011). The influence of writing tools and mechanics on text production. Vortrag in der Arbeitsgruppe "Prozesse der Textproduktion in der Schule: Strukturen und Verarbeitung aus sprachdidaktischer und psycholinguistischer Perspektive" bei der DGfS Jahrestagung 2011.

Michael Piotrowski, Cerstin Mahlow und Robert Dale (Hg.) (2010). *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics and Writing: Writing Processes and Authoring Aids*. Stroudsburg, PA, USA: Association for Computational Linguistics.

Curriculum Vitae

	Cerstin Elisabeth Mahlow, M.A.
Adresse	Käshaldenstrasse 1
	8052 Zürich
Geboren am	31. Oktober 1975
Geboren in	Greifswald, Deutschland

Akademische Laufbahn und Berufserfahrung

Seit 11/2010	**Wissenschaftliche Mitarbeiterin** im SNF-Projekt *Sprichwörter und Redewendungen im Wandel. Online-Lexikon zur diachronen Phraseologie* am Deutschen Seminar der *Universität Basel*, Basel (Prof. Dr. Annelies Häcki Buhofer)
12/2008 – 12/2010	**E-Learning-Beauftragte, Wissenschaftliche Mitarbeiterin** an der *Hochschule für Soziale Arbeit Fachhochschule Nordwestschweiz*, Olten/Basel
01/2007 – 06/2007	**Mitarbeiterin** im IIL-Projekt *Dynamische Erstellung personalisierter Lehrmaterialien* am Institut für Computerlinguistik der *Universität Zürich*, Zürich (Prof. Dr. Michael Hess)
10/2004 – 11/2008	**E-Learning-Koordinatorin** der Philosophischen Fakultät der *Universität Zürich*, Zürich
01/2005 – 12/2006	**Mitarbeiterin** im SVC-Projekt *TransTech – Language Technology for Translators* am Institut für Computerlinguistik der *Universität Zürich*, Zürich (Prof. Dr. Michael Hess)

10/2003 – 06/2004	**Freie Mitarbeiterin** bei *imachine projekt ag*, Zürich
seit 2003	**Lehrbeauftragte** der Philosophischen Fakultät der *Universität Zürich*, Zürich
04/2001 – 03/2003	**Wissenschaftliche Assistentin** im Rahmen eines ICT-Projektes am Institut für Computerlinguistik der *Universität Zürich*, Zürich (Prof. Dr. Michael Hess)

Ausbildung

10/2003	**E-Learning-Zertifikat** der ETH, Universität und Pädagogischen Hochschule Zürich, 15 ECTS
2002 – 2003	**PRO-WISS**-Kurse der Universität Zürich im Bereich Selbst- und Projektmanagement sowie zum Führen im akademischen Umfeld
02/2001	**Magistra Artium** in Linguistischer Informatik, Iberoromanistik und Politikwissenschaften, *Friedrich-Alexander-Universität Erlangen-Nürnberg*, Erlangen, Note 1,7 (gut) **Magisterarbeit** *Automatische Wortformanalyse für das Spanische* Gutachter: Prof. Dr. Roland R. Hausser, Prof. Dr. Jürgen Lang.
10/1995–02/2001	**Magisterstudium** der Linguistischen Informatik, Iberoromanistik und Politikwissenschaften, *Friedrich-Alexander-Universität Erlangen-Nürnberg*, Erlangen
1995	**Abitur** am *Städtischen Gymnasium Prenzlau*, Prenzlau, Note 1,1 (sehr gut)

Zürich, den 24. Juli 2011

Cerstin Mahlow